ADRIAN ATANASIU

CURS DE LINGVISTICĂ MATEMATICĂ

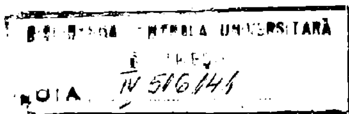
Editura Universității din București
1998



ADRIAN ATANASIU

CURS
DE
LINGVISTICĂ MATEMATICĂ

EDITURA UNIVERSITĂŢII DIN BUCUREŞTI
- 1998 -



Referenţi ştiinţifici: **Lector dr. Mihaela MALIŢA**
Lector dr. Victor MITRANA

Şos. Panduri 90-92, Bucureşti - 76235; Tel./Fax: 410.23.84



ISBN 973 – 575 – 270 – 0

Prefaţă

Tradiţia şcolii româneşti de lingvistică matematică (*Computational linguistics*) are o vechime de circa trei decenii, de la lucrările profesorului Solomon Marcus despre gramatici transformaţionale. Ea s-a dezvoltat continuu, obţinând un prestigiu unanim recunoscut prin lucrările de limbaje formale şi aplicaţii ale acestora; domenii cum ar fi *Sisteme de gramatici*, *Artificial life*, *Splicing* sunt fundamentate de cercetători din şcoala matematică românească. Studii de tratare matematică a limbajului natural în general, a limbii române în particular au fost mai puţine şi s-au elaborat cu precădere la Institutul de cercetări al Academiei Române. Începând cu 1995 ele au captat interesul unui colectiv de la Facultatea de Matematică a Universităţii Bucureşti, acesta alăturându-se cercetărilor similare efectuate la Universităţile din Iaşi şi Cluj.

Lucrarea de faţă reprezintă suportul unui curs de un semestru ţinut la studenţii care urmează ciclul de studii aprofundate. Scopul său principal este de a face o descriere "pe primul nivel" a teoriilor şi tehnicilor folosite în domeniul prelucrării limbajului natural.

Studiul limbajului natural face apel la cunoştinţe din numeroase alte domenii de cercetare, cum ar fi matematica, lingvistica, logica, psihologia (în relaţia sa cu dezvoltarea limbajului). Foarte puţini oameni pot acoperi simultan toate aceste domenii; de aceea cunoştinţele introduse aici rămân aproape permanent la un nivel informativ, de foarte puţine ori realizând un studiu complet.

Structura cursului urmăreşte în linii mari lucrarea *Natural Language Understanding* a lui James Allen de la Universitatea Rochester; am introdus însă şi capitole noi (renunţând la altele), am încercat permanent (unde a fost posibil) folosirea exemplelor în limba română, am modificat centrul de greutate în partea sintaxei operând cu noţiuni pe care studenţii le cunosc din anii anteriori.

Există bineînţeles elemente care nu au fost abordate. De exemplu, nu am tratat deloc ideea de *speech recognition*, deşi în acest moment ea este în plin centru de interes. De asemenea, nu am intrat în detalii tehnice despre cum se construieşte un corpus (de care limba română duce încă lipsă), cum se rezolvă probleme de ambiguitate, metafore, analiza statistică a textelor, etc. Ele trebuiau să fie prezentate, dar lucrarea a fost gândită ca un suport de curs pentru un semestru. Atunci când interesul pentru studiul limbajului natural va creşte în aşa măsură încât să poată fi acceptat un curs de două semestre, poate va fi scoasă şi o completare a sa.

Conf. Dr. Adrian Atanasiu

e-mail: adrian@matem.buc.soros.ro

Prelegerea 1

Studiul limbajului

1.1 Ce este un limbaj ?

Limbajul este unul din aspectele fundamentale ale comportamentului uman, element crucial al vieţii. În formă scrisă el serveşte ca înregistrare pe termen lung, util transmiterii cunoştinţelor de la o generaţie la alta. În formă vorbită, el este una din coordonatele primare ale relaţiei dintre oameni.

Din punct de vedere ştiinţific, limbajul poate fi studiat de diverse discipline (vezi Tabelul 1.1); fiecare din ele defineşte propriul ei set de probleme şi are modul ei propriu de adresare.

- **Lingvistica** de exemplu, studiază structura limbajului brut, punând întrebări de genul *De ce anumite combinaţii de cuvinte formează propoziţii iar altele nu*, *De ce o propoziţie poate avea înţeles iar altele nu*, etc.

- **Psiholingviştii** pe de altă parte, studiază procesele producerii şi înţelegerii limbii naturale, luând în considerare întrebări cum ar fi *Cum identifică oamenii structura corectă a unei propoziţii*, sau *Când se decide înţelesul corect al cuvintelor.*

- **Filozofii** îşi îndreaptă atenţia asupra modului în care cuvintele ajung să semnifice obiecte - şi cum sunt identificate obiectele prin cuvinte. De asemenea ei studiază ce înseamnă să ai certitudini, scopuri şi intenţii şi cum se leagă de limbaj aceste capacităţi cognitive.

- Scopul **lingvisticii matematice** este de a dezvolta o teorie computaţională a limbajului, folosind elemente ale informaticii (algoritmi, structuri de date, etc). Bineînţeles, în construcţia unui model computaţional trebuie folosit în mod eficient ceea ce se ştie deja de la celelalte discipline.

Deci se pot desprinde două motivaţii pentru dezvoltarea modelelor computaţionale:

1. **Motivaţia ştiinţifică** are rolul de a obţine o înţelegere mai bună asupra modului cum lucrează limbajul. Se ştie că nici una din celelalte discipline tradiţionale nu dispune de echipament suficient de dezvoltat pentru a trata

această problemă în totalitate. Chiar dacă se va proceda la o combinare a tuturor ipotezelor dezvoltate, o teorie completă ar fi mult prea complexă pentru a putea fi studiată cu metodele tradiţionale. În schimb, aceste teorii complexe pot fi modelate algoritmic, programate şi apoi testate pe calculator. În funcţie de performanţele oferite de acest mod de simulare, se pot trage diverse concluzii.

Modelele computaţionale pot oferi idei deosebit de utile referitoare la comportamentul lingvistic (şi deci uman) în viitor, idei care pot fi explorate de psiholingvişti. Continuând acest proces, se poate eventual obţine o mai bună înţelegere asupra modului cum acţionează limbajul uman. Acest proiect ocupă multe eforturi unite ale lingviştilor, psiholingviştilor, filozofilor şi informaticienilor, creind o nouă zonă de cercetare interdisciplinară, numită de obicei *cognitive science*.

2. **Motivaţia practică** (sau **tehnologică**) enunţă ideea că, datorită capacităţilor sale, utilizarea limbajului natural va revoluţiona modul de folosire al calculatoarelor. Deoarece majoritatea cunoştinţelor umane sunt păstrate sub formă lingvistică, computerele care vor putea înţelege limbajul natural vor putea avea acces la toată această informaţie. În plus, o interfaţă de limbaj natural cu calculatoarele va putea să le facă pe acestea accesibile tuturor. Astfel de sisteme vor fi considerabil mai flexibile şi mai inteligente decât este posibil cu tehnologia curentă. Practic, nu este important dacă modelul folosit reflectă modul uman de înţelegere al limbajului. Important este dacă el funcţionează.

În general, în dezvoltarea unei teorii a prelucrării limbajului natural se abordează elemente din ambele motivaţii. Aceasta, datorită pe de-o parte ideii că un limbaj natural este atât de complex încât o modelare ad-hoc, fără o specificare teoretică precisă nu va fi eficientă. Deci scopul tehnologic nu poate fi realizat fără o bază teoretică sofisticată la nivelul celor dezvoltate de lingvişti, psiholingvişti şi filozofi. Pe de-altă parte, starea actuală a cunoştinţelor despre prelucrarea limbajului natural este abia la început, astfel încât orice încercare de a construi un model cognitiv coerent nu este fezabilă.

Suntem încă în faza în care încercăm orice model practic care pare că merge.

1.2 Aplicaţii ale înţelegerii limbajului natural

O modalitate de a face o introducere în prezentarea studiilor relative la limbajul natural este de a trece în revistă diverse aplicaţii la care lucrează sau au lucrat cercetătorii din domeniu. Aceste aplicaţii se pot împărţi în două clase mari:

1. Aplicaţii bazate pe text;

2. Aplicaţii bazate pe dialog.

Aplicaţiile *bazate pe text* se referă la prelucrarea textului scris (cărţi, reviste, rapoarte, mesaje e-mail etc). Toate sunt tehnici referitoare la probleme de citire. Cercetările limbajului natural bazate pe text conduc la aplicaţii cum ar fi:

Tabelul 1.1:

Disciplină	**Probleme specifice**	**Metode folosite**
Lingvistică	*Cum formează cuvintele fraze şi afirmaţii ? Care sunt restricţiile înţelesurilor unei afirmaţii ?*	*Intuiţii despre forma corectă şi semnificaţie; modele matematice asupra structurii.*
Psiho-lingvistică	*Cum identifică oamenii structura frazei ? Cum sunt identificate înţelesurile cuvintelor ? Când are loc procesul de înţelegere ?*	*Tehnici experimentale bazate pe măsurarea performanţelor umane; analiza statistică a observaţiilor.*
Filosofie	*Ce este înţelesul şi cum este el asociat cuvintelor şi propoziţiilor ? Cum identifică cuvintele obiectele din spaţiul înconjurător ?*	*Argumentaţii ale limbajului natural bazate pe intuiţie şi exemple; modele matematice (logică, teoria modelelor).*
Lingvistică computaţională (matematică)	*Cum este identificată structura propoziţiilor ? Cum se pot modela raţionamentul şi cunoaşterea ? Cum poate fi folosit limbajul pentru a realiza aceste obiective ?*	*Algoritmi, structuri de date şi modele formale ale raţionamentului; tehnici de inteligenţă artificială.*

- Aflarea documentelor pe o anumită temă dintr-o bază de date de texte (de exemplu, aflarea cărţilor semnificative dintr-o bibliotecă);

- Extragerea informaţiei utile din mesaje sau articole dintr-un anumit domeniu (de exemplu, construcţia unei baze de date despre toate tranzacţiile de aur anunţate în ştirile unei zile);

- Translatarea documentelor dintr-o limbă în alta (de exemplu, reperele unei mărci de maşină, în mai multe limbi diferite);

- Rezumate ale textelor într-un anumit domeniu (de exemplu, realizarea unui sumar de 3 pagini dintr-un raport guvernamental de 1000 pagini).

Nu toate modelele care realizează aceste obiective folosesc tehnici de înţelegere a limbajului natural (în sensul care va fi dezvoltat în curs). Să ne referim de pildă la determinarea articolelor dintr-un anumit domeniu dintr-o bază mare de date. Există în această direcţie mulţi algoritmi care clasifică documentele după prezenţa în text a anumitor cuvinte cheie. Se pot găsi articole dintr-un domeniu căutând articolele care conţin cuvintele cheie asociate domeniului respectiv. Articolele de drept de exemplu, pot conţine cuvinte cum ar fi *apel*, *curte*, *recurs*, *înfăţişare*, în timp ce

articolele despre comerţ pot conţine cuvinte specifice cum ar fi: *tranzacţie, piaţă, marfă, preţ*. Un astfel de algoritm poate găsi articole din orice domeniu care are predefinit un set de cuvinte cheie.

Articolele ştiinţifice din ultima vreme apar în revistele de specialitate însoţite de astfel de cuvinte cheie, în scopul unei rafinări a căutării pe domenii.

Evident, nu se poate spune că aceşti algoritmi conduc la înţelegerea textului; sunt simple tehnici de căutare a unor cuvinte în text. Deşi conduc la aplicaţii utile, ele sunt inerent limitate.

Astfel, nu putem extinde căutarea la texte al căror subiect este exprimat printr-un rezumat, cum ar fi de exemplu:

Să se găsească toate problemele apărute între* 1980 – 1985 *a căror rezolvare s-a redus la problema celor patru culori.

Pot exista astfel de probleme care să nu fi folosit cuvintele cheie respective, care să nu fie de matematică, etc.

Pentru a putea face faţă unor asemenea cerinţe extinse, sistemul folosit trebuie să fie capabil să extragă suficient de multă informaţie din fiecare articol din baza de date, pentru a putea decide dacă acel articol satisface sau nu cerinţele de căutare; deci trebuie să construim o reprezentare a informaţiei din articol şi apoi să folosim această reprezentare în scopuri de căutare. Aceasta exprimă o cerinţă majoră a unui sistem de înţelegere: el trebuie să calculeze o anumită reprezentare a informaţiei, pe care să o folosească ulterior în algorimi de inferenţă.

Să considerăm alt exemplu. Anumite sisteme automate de translatare au fost construite pe ideea comparării unor structuri-tip (pattern): o secvenţă de cuvinte dintr-un limbaj este asociată unei secvenţe de cuvinte din alt limbaj. Translatarea este realizată prin găsirea celui mai bun set de structuri-tip care corespunde intrării. Această tehnică (folosită mai ales în traduceri automate de texte dedicate) poate produce rezultate rezonabile în anumite situaţii, dar poate conduce şi la translatări complet greşite în cazul unor construcţii cu înţeles apropiat, pe care nu este abilitată să le poată înţelege.

Alţi algoritmi lucrează în doi paşi: produc întâi o reprezentare a înţelesului fiecărei propoziţii într-un limbaj, iar apoi determină o propoziţie în alt limbaj, cu acelaşi înţeles. Rezultatul final are o corectitudine mai mare, afirmaţie bazată pe o analiză semantică care foloseşte tehnici computaţionale de înţelegere a limbajului natural.

Un domeniu foarte atractiv pentru cercetările bazate pe text este acela de *înţelegere a povestirii*. Aici, sistemul prelucrează o povestire, după care trebuie să răspundă la întrebări relative la ea. Aceasta este similară testelor folosite în şcoli referitoare la înţelegerea citirii.

Aplicaţiile *bazate pe dialog* folosesc comunicarea om - maşină. Ca bază normală de lucru se utilizează limbajul vorbit, dar se poate folosi ca interfaţă şi tastatura.

Exemple de aplicaţii practice:

- *Sisteme de răspuns* la întrebări (limbajul natural foloseşte la interogarea unei baze de date);

- *Serviciu telefonic automat* (pentru a realiza de exemplu anumite operaţii bancare sau a cere anumite obiecte dintr-un catalog);
- *Sisteme de seminarizare* (maşina *conversează* cu elevul pe baza unei anumite lecţii);
- *Sisteme de cooperare* în rezolvarea unor probleme (de exemplu algoritmi care ajută la planificarea curselor aviatice; sau în arhitectură, la simularea proiectelor).

De obicei aceste probleme sunt tratate prin sistemele bazate pe dialog într-un mod complet diferit faţă de tratarea în sistemele bazate pe text. Limbajul folosit este altul, iar sistemul cere o participare activă pentru a menţine un dialog natural, uşor de urmărit. Dialogul reclamă atât folosirea de cunoştinţe pentru a verifica dacă lucrurile sunt înţelese, cât şi abilitate în recunoaşterea şi generarea de subdialoguri pentru clarificarea anumitor concepte. Dar chiar şi cu aceste diferenţe, algoritmii de bază sunt fundamental aceiaşi.

Este important să separăm problemele de recunoaştere a vorbirii de cele de înţelegere a limbajului. Un sistem de recunoaştere a vorbirii nu foloseşte nici un element de înţelegere a limbajului. El este utilizat numai pentru identificarea cuvintelor rostite cu o anumită claritate într-o limbă fixată, nu şi pentru înţelegerea mesajului pe care îl conţin acestea. Pentru a fi un sistem de înţelegere a limbajului, un sistem de recunoaştere a vocii trebuie completat la intrare cu un sistem de înţelegere a limbajului natural, ceea ce conduce în final la ceea ce numim *sistem de înţelegere a limbajului vorbit.*

1.3 Sisteme de evaluare a înţelegerii limbajului

După cum s-a văzut, ceea ce contează ca înţelegere poate varia de la o aplicaţie la alta. Se poate pune atunci întrebarea:

De unde ştim când este bun sistemul ?

O modalitate imediată de evaluare constă în rularea programului şi compararea rezultatelor cu ceea ce ne aşteptăm să obţinem. Este ceea ce se numeşte ***black box evaluation*** - deoarece apreciem performanţele sistemului fără să cercetăm modul în care acesta ajunge la ele. Folosită ca metodă finală de evaluare, ea poate fi considerată ca fiind cel mai bun test de validare; în schimb nu se poate folosi în etapele de început ale cercetării, pentru că aici rezultatele pot duce la aprecieri greşite. În general tehnicile care produc rezultate bune pe termen scurt nu vor obţine aceleaşi performanţe pe termen lung.

O altă metodă de evaluare constă în identificarea diverselor subcomponente ale sistemului şi testarea lor separată. Această strategie poartă numele de *glass box evaluation*, deoarece permite observarea structurii interne a sistemului. Dificultatea acestui mod de evaluare constă în faptul că sunt necesare anumite coerenţe (sau consensuri) între pachetele de teste. Relaţiile care asigură asemenea consensuri stau la baza unor teorii noi relative la prelucrarea limbajului natural.

În ciuda acestor dificultăţi, evaluarea are o importanţă crucială. Se ştie că un număr mic de teste aplicate unui anumit sistem nu conduce la o concluzie clară. Dacă utilizatorul nu lucrează direct cu sistemul ci prin intermediul mai multor interfeţe, rezultatele pot fi afectate într-un procent care nu se poate neglija.

Un alt pericol este acela că utilizatorii acordă în mod reflex mai multă inteligenţă programului, decât este acesta abilitat. Să exemplificăm această afirmaţie.

La mijlocul anilor '60, **MIT** a dezvoltat un program numit **ELIZA**, care a devenit unul din cele mai populare programe de inteligenţă artificială din toate timpurile. ELIZA nu a pretins niciodată că se bazează pe o teorie a înţelegerii limbajului, dar serveşte ca un excelent exemplu în această direcţie. Sistemul joacă rolul unui medic psiholog şi, pentru a obţine cele mai bune rezultate, utilizatorul va juca rolul unui pacient. În Tabelul 1.2 dăm exemplul unui dialog posibil între cei doi parteneri.

Tabelul 1.2:

Utilizator	**ELIZA**
Toţi oamenii sunt nefericiţi.	*De ce ?*
Totdeauna când au o problemă, deranjează pe alţii.	*Poţi să dai un exemplu ?*
Da, ieri a venit la mine un prieten.	*Un prieten a venit ieri la tine.*
Spunea că sufăr de stres de mai mult timp.	*Îmi pare rău să aud că suferi de stres.*
Si este adevărat, sufăr de stres.	*Crezi că venind aici vei fi ajutat ?*
Am nevoie de ajutor; asta este sigur.	*Ce înţelegi prin ajutor ?*
Poate să învăţ să fiu mai apropiat de ai mei.	*Spune-mi mai multe despre familia ta.*
Mama are grijă de mine.	*Cine mai are grijă de tine în familie ?*
Tata.	*Tata.*
Într-un fel, semeni cu tatăl meu.	*Ce asemănare vezi ?*

Să vedem cum lucrează *ELIZA*.

Există o bază de date formată din cuvinte cheie. Pentru fiecare cuvânt cheie, sistemul stochează un număr întreg (prioritate), o structură - tip pe care o va întâlni la intrare şi o specificare a ieşirii. Algoritmul este următorul:

Fiind dată o propoziţie α, se caută un cuvânt cheie în α, a cărui structură - tip corespunde cu α. Dacă în α sunt mai multe cuvinte cheie, se alege cel care are cea mai mare prioritate. Se folosesc specificaţiile de ieşire pentru a da răspunsul.

Dacă în α nu sunt cuvinte cheie, se generează o propoziţie de continuare, cum ar fi

Spune-mi mai multe despre ... sau
Continuă.

În Tabelul 1.3 este un fragment dintr-o bază de date a cuvintelor cheie. O structură - tip constă din cuvinte şi variabile. Prefixul ? din faţa unei litere indică o variabilă, care poate fi înlocuită cu orice secvenţă de cuvinte. De exemplu, structura-

ţip

?X este ?Y

va coincide cu propoziţia

Cine este persoana care a venit ieri ?

Aici variabila *?X* este înlocuită cu *Cine* iar *?Y* cu *persoana care a venit ieri ?*. Specificaţiile de ieşire pot folosi aceleaşi variabile. *ELIZA* inserează valorile variabilelor date de structura de intrare în cea de ieşire, după ce a făcut eventual mici modificări de pronume (a înlocuit *eu* cu *tu*, *mine* cu *tine* etc). Deci, dacă pentru structura - tip de mai sus, specificaţia de ieşire este

Cunoşti ?Y ?

atunci pentru intrarea

Cine este persoana care a venit ieri ?

se va genera răspunsul

Cunoşti persoana care a venit ieri ?

Tabelul 1.3:

Cuvânt	**Prioritate**	**Structură-tip**	**Ieşire**
Nefericit	10	*?X*	*De ce ?* *În ce fel ?*
este	3	*?X este ?Y*	*Cunoşti ?Y ?*
totdeauna	5	*?X*	*Poţi să dai un exemplu ?* *Când ?* *Într-adevăr, totdeauna ?*
ce	2	*?X*	*De ce întrebi ?* *Te interesează ?*

Dacă pentru o structură -tip sunt mai multe ieşiri posibile, *ELIZA* selectează câte una distinctă pentru fiecare apariţie, în scopul de a preveni repetiţia. Primele două răspunsuri ale *ELIZEI* din Tabelul 1.2 sunt generate pe baza acestei baze de date.

După cum se vede din descrierea algoritmului *ELIZA*, programul nu înţelege conversaţia la care participă. Practic, el este o colecţie de şiretlicuri.

De ce totuşi a avut *ELIZA* un succes atât de mare ? Pot fi mai multe răspunsuri posibile. Cel mai probabil este acela că, atunci când oamenii aud sau citesc o secvenţă de cuvinte pe care o înţeleg ca o propoziţie, ei atribuie înţelesul respectiv acelei propoziţii şi consideră automat că el aparţine celui care a produs-o (chiar dacă este o maşină). Deci *ELIZA* apare ca inteligentă pentru că interlocutorii ei îşi folosesc propria lor inteligenţă pentru a da un sens propoziţiilor.

Iluzia de inteligenţă este susţinută şi de o altă caracteristică. Sistemul nu are nevoie de nici un cuvânt inteligent, pentru că el nu face afirmaţii, nu argumentează şi nu răspunde la întrebări. Pur şi simplu, el pune o serie de întrebări, ceea ce, înafara situaţiei pacient - doctor, ar fi o discuţie inacceptabilă. *ELIZA* ocoleşte toate întrebările răspunzând cu alte întrebări, cum ar fi în ultima instanţă

De ce întrebi ?.

Nu există nici o posibilitate în a face programul să spună ceva concret despre orice.

Chiar şi în aceste situaţii restrictive, este relativ uşor să demonstrăm că programul nu înţelege. El produce uneori răspunsuri complet aiurea. De exemplu, pentru *ELIZA* pe baza căruia s-a dat exemplul din Tabelul 1.2, afirmaţia

Repetiţia este mama înţelepciunii.

va genera (datorită cuvântului cheie *mama*) ieşirea

Spune-mi mai multe despre familia ta.

De asemenea, cu cât conversaţia progresează, devine evident că *ELIZA* nu reţine nimic din conţinutul discuţiei; ea va începe să pună întrebări care sunt fără sens în lumina discuţiei anterioare.

Să presupunem că vi se cere ca în şase luni să construiţi un program de limbaj natural pentru o anumită aplicaţie. Dacă veţi începe construcţia unui model general al înţelegerii limbajului, el sigur nu va fi gata în intervalul de timp considerat şi va obţine rezultate dezastruoase la teste. Un sistem tip *ELIZA* care să producă un comportament de tipul celui de sus, poate fi scris în mai puţin de o jumătate de an; şi cu siguranţă că el va trece cu succes multe teste. Deci, dacă luăm ca unic criteriu performanţa pe termen scurt, un program tip *ELIZA* va fi preferat; acesta este totuşi inacceptabil pentru scopul final al proiectului.

Pentru a elimina această posibilitate, va trebui sau să acceptăm anumite ipoteze teoretice despre arhitectura sistemelor limbajului natural şi să dezvoltăm teste de evaluare specifice diferitelor componente, sau să ignorăm rezultatele slabe ale testelor de evaluare până când sistemul ajunge la un nivel rezonabil de ridicat de performanţă.

1.4 Nivele ale analizei limbajului

Un program de limbaj natural trebuie să se bazeze pe cunoştinţe considerabile despre structura limbajului însuşi, inclusiv ce sunt cuvintele, cum se combină ele pentru a forma propoziţii, ce înţeles au cuvintele, cum contribuie înţelesul lor la înţelesul propoziţiei ş.a.m.d. Totuşi aspectul lingvistic al problemei, deşi necesar, nu este şi suficient; intervine şi un studiu al comportamentului uman - cunoştinţe generale despre lumea înconjurătoare, precum şi abilităţi de raţionament. De exemplu, pentru a răspunde la întrebări şi a participa la o conversaţie, o persoană trebuie să cunoască nu numai structura limbajului folosit, dar să se bazeze şi pe o cultură personală.

Nivelurile de cunoştinţe relevante pentru înţelegerea limbajului natural pot fi clasificate în felul următor:

1. **Nivelul fonetic şi fonologic**; se referă la modul de exprimare al cuvintelor prin sunete (modul de legătură şi de realizare cuvânt - sunet). Astfel de cunoştinţe sunt cruciale în sistemele bazate pe voce.

2. **Nivelul morfologic**; se referă la modul de construcţie al cuvintelor din o serie de elemente de bază numite *morfeme*. Un morfem este unitatea structurală de bază din limbaj care are asociat un înţeles (De exemplu, înţelesul cuvântului

prietenos este derivat din înţelesul substantivului *prieten* şi a sufixului *os*, care transformă un substantiv într-un adjectiv).

3. **Nivelul sintactic**; tratează modul de combinare al cuvintelor pentru a forma propoziţii corecte, determină rolul structural al fiecărui cuvânt în propoziţie precum şi relaţia propoziţiilor într-o frază.

4. **Nivelul semantic**; se referă la semnificaţia cuvintelor, modul de combinare al acestor semnificaţii pentru a forma semnificaţia unei propoziţii. Este studiul unui înţeles independent de context: o propoziţie tratată fără nici o legătură cu contextul în care a fost utilizată.

5. **Nivelul pragmatic**; tratează folosirea propoziţiilor în diverse situaţii, modul în care influenţează contextul interpretarea unei propoziţii.

6. **Nivelul discursului**; se ocupă de modul în care propoziţiile anterioare afectează interpretarea propoziţiilor care urmează. Un aspect important se referă la interpretarea pronumelor şi la aspectul temporal al informaţiei vehiculate.

7. **Nivelul cunoaşterii universului**; include cunoştinţe generale despre structura lumii la care face referire utilizatorul pentru ca, de exemplu, să facă faţă conversaţiei. Sunt incluse toate informaţiile pe care trebuie să le aibă un utilizator despre partenerii săi de discuţie.

Evident, aceasta este o clasificare destul de imprecisă, care nu acoperă chiar toate preocupările referitoare la lingvistică. Orice eveniment particular poate include aspecte din mai multe nivele, şi un algoritm va trebui să trateze simultan nivele diferite.

Exemplul 1.1 *Să încercăm să arătă m prin exemple diferenţele dintre sintaxă, semantică şi pragmatică. Fie următoarele propoziţii:*

1. *Broaştele sunt fiinţe.*

2. *Broaştele verzi au nasuri mari.*

3. *Ideile verzi au nasuri mari.*

4. *Mari ideile verzi are nasuri.*

Prima propoziţie poate fi considerată corectă sintactic, semantic şi pragmatic. Fiecare din celelalte afirmaţii violează însă unul sau mai multe din aceste concepte.

Propoziţia 2 este corectă sintactic şi semantic dar nu şi pragmatic. Cunoştinţele noastre despre broaşte ar respinge valoarea ei de adevăr.

Cu propoziţia 3 este mai rău. Ea este corectă sintactic dar este greşită semantic şi pragmatic. Dacă propoziţia 2 poate fi lansată într-o discuţie şi combătută eventual cu argumente, propoziţia 3 nu poate fi nici măcar enunţată într-o conversaţie coerentă. Ea nu se poate nici afirma şi nici nega. Un program care să o nege pragmatic, ar trebui să demonstreze că ideile nu pot fi verzi sau - dacă sunt - în mod sigur nu au nasul mare.

Propoziţia 4 *nu este corectă din nici unul din cele trei puncte de vedere (sintactic, semantic, pragmatic). Ea este de fapt neinteligibilă; nu are o structură pe baza căreia să putem decide ce este greşit.*

Este interesant că se pot găsi exemple în care o propoziţie, să fie greşită sintactic deşi este corectă pragmatic. Dacă te întreabă cineva unde mergi şi răspunzi

Eu merg facultate.

deşi este greşită sintactic, ea are un înţeles pragmatic corect şi - eventual, este chiar şi semantic corectă.

1.5 Reprezentări şi înţelegeri

O componentă crucială a înţelegerii foloseşte calculul reprezentării înţelesului propoziţiilor şi textelor. Chiar şi fără o definire a noţiunii de reprezentare, această afirmaţie este neclară. De exemplu, de ce nu se foloseşte propoziţia însăşi ca o reprezentare a înţelesului ei ? Un motiv este acela există multe cuvinte cu înţelesuri multiple, pe care le numim *sensuri*. Cuvântul *port* de exemplu are un sens de substantiv (loc unde acostează navele) şi sens de verb (a purta, persoana 1 singular, prezent). Astfel de ambiguităţi ar crea multiple dificultăţi sistemului. Pentru oameni, rezolvarea lor este foarte simplă, reieşind din context; de multe ori oamenii nu observă ambiguităţile decât dacă li se atrage atenţia, pentru amuzament. Sunt celebre versurile lui Eminescu

Să -ţi fie somnul lin sau

Cobori în jos luceafăr blând

cărora li se pot găsi astfel de ambiguităţi în exprimare.

Dar dacă o persoană nu pare să ia în considerare toate sensurile unui cuvânt, acest lucru nu este posibil în cazul unui program care trebuie să le trateze explicit.

Pentru a reprezenta înţelesul, este necesar un limbaj mai precis. Acesta este oferit de *Limbaje formale*, de *Logică* şi de *Matematică.* O reprezentare utilă a limbajului satisface următoarele două proprietăţi:

- Este precisă şi neambiguă. Ea va permite exprimarea oricărei citiri a unei propoziţii ca o formulă distinctă.
- Reprezentarea trebuie să păstreze structura intuitivă a propoziţiilor limbajului natural pe care îl modelează. De exemplu, propoziţiile cu structuri similare trebuie să aibă reprezentări similar structurate; sau - înţelesurile a două propoziţii care sunt parafraze una alteia, trebuie să fie foarte apropiate.

1.5.1 Sintaxă; Structura de reprezentare a propoziţiilor

Structura sintactică a unei propoziţii indică modul în care sunt legate între ele cuvintele propoziţiei. Ea arată cum sunt grupate cuvintele, cum anumite cuvinte modifică altele şi care cuvinte sunt de importanţă centrală în frază. În plus, sintaxa poate identifica tipurile de relaţii care există între fraze şi poate reţine diverse informaţii despre structura propoziţiei, care vor fi necesare în prelucrarea ulterioară .

Exemplul 1.2 *Să considerăm următoarele două propoziţii:*

1. *Ion îi vinde lui Maria cartea.*
2. *Cartea este vândută Mariei de către Ion.*

Propoziţiile au anumite similitudini structurale: ambele folosesc substantivele Ion, Maria, carte; ambele descriu o acţiune de vânzare. Din alte puncte de vedere, propoziţiile sunt semnificativ deosebite. Chiar dacă ambele au aceeaşi valoare de adevăr, la întrebarea

Ce face Ion pentru Maria ?

nu poate fi folosit ca răspuns propoziţia 2, *ci numai* 1.

Propoziţia 2 *poate fi folosită ca o continuare a unei propoziţii de tipul* **Deşi este ruptă...**, *ca în cele ce urmează:*

3. *(*) Deşi este ruptă, Ion îi vinde lui Maria cartea.*
4. *Deşi este ruptă, cartea este vândută Mariei de către Ion.*

În continuare se va folosi notaţia (*) pentru orice exemplu de propoziţie incorectă sau problematic corectă.

Multe alte proprietăţi pot fi relevate considerând propoziţii greşit formulate. Propoziţia 5 *este greşită datorită dezacordului subiect- predicat. Propoziţia* 6 *este greşită deoarece verbul* pune *solicită o informaţie suplimentară (*ce pune ?*).*

5. *(*) Ion sunt în colţ.*
6. *(*) Ion pune pe masă.*

Analiza gramaticală a unui text nu constituie scopul final al unui model de înţelegere a limbajului natural. De fapt, un bun sistem trebuie să fie capabil să înţeleagă, ori de câte ori poate, propoziţiile prost formate. Astfel, între propoziţia 5 *şi*

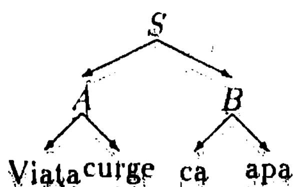
5'. *Ion este în colţ.*

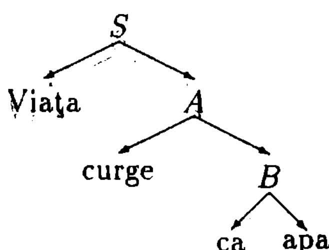
nu se face nici o distincţie.

Multe reprezentări sintactice ale unui limbaj folosesc modalitatea arborescentă oferită de gramaticile independente de context. De exemplu, pentru propoziţia

Viaţa curge ca apa

se pot da două reprezentări sintactice distincte (în funcţie de gramatica care generează aceste propoziţii):





1.5.2 Forma logică

Structura unei propoziţii nu reflectă şi înţelesul ei. De exemplu, substantivul *broască* poate avea diverse înţelesuri, în funcţie de propoziţia în care este folosit. Ce semnificaţie se poate da propoziţiei

Emil are o broască ?

Aceea că Emil are o fiinţă (*broască*) sau un obiect pe care urmează să-l monteze la uşă (*broască*). Propoziţia este corectă sintactic, semantic şi pragmatic, dar ambiguă din punct de vedere al semnificaţiei. Să presupunem că din discuţie reiese că a fost vorba de primul înţeles. Nici acuma semnificaţia nu este completă, putând fi vorba de diverse specii de broaşte.

Înţelesul unei propoziţii depinde de situaţia în care este folosită propoziţia respectivă. Există aici o relaţie între dependenţa de context şi independenţa de context. Faptul că *broască* se referă la un lucru sau o fiinţă, este o informaţie legată de limba română, independent de locul în care este folosit cuvântul. Pe de-altă parte substantivul broască se referă la ceva ce are Emil, şi depinde de contextul conversaţiei.

Definiţia 1.1 *Reprezentarea înţelesului independent de context al unei propoziţii se numeşte formă logică.*

Forma logică codifică sensuri diverse ale cuvintelor şi identifică relaţiile semantice între cuvinte şi fraze. Multe astfel de relaţii sunt regăsite în relaţiile semantice dintre verb şi substantivul asociat.

De exemplu, în propoziţiile 1 şi 2, acţiunea descrisă este una de vânzare, unde *Ion* este vânzătorul, *cartea* este obiectul în cauză, iar *Maria* este cumpărătorul. Aceste roluri sunt instanţieri ale unor obiecte semantice generale, să le numim

$$AGENT, TEMA, POSESOR - FINAL.$$

Odată determinate relaţiile semantice, unele sensuri ale cuvintelor pot deveni imposibile, şi deci vor fi eliminate dintre posibilităţi.

Exemplul 1.3 *Să luăm propoziţia:*

Mama ta de-i vie, bine-ar fi să vie pe la noi la vie.

Cuvântul vie *este folosit aici de trei ori, de fiecare dată cu altă semnificaţie. Determinarea interpretării semantice se face însă în mod unic, datorită legării lui de semnificaţiile celorlaltor cuvinte. Astfel, prima utilizare este legată de cuvântul* **mama** *şi devine adjectiv; a doua apariţie este în relaţie tot cu* **mama**, *dar prin poziţia sa i se asigură interpretarea semantică de verb. În sfârşit, fiind legat de predicat, a treia apariţie a cuvântului* vie *este complement de loc, cu semnificaţie de substantiv.*

Una din problemele cheie în interpretarea semantică este aceea de a stabili ce semnificaţii individuale ale cuvintelor se pot combina pentru a crea un înţeles coerent al propoziţiei. Prin exploatarea interconexiunilor dintre semnificaţiile cuvintelor, se poate reduce masiv numărul sensurilor pe care le poate avea fiecare cuvânt în propoziţie.

1.5.3 Reprezentarea înţelesului final

Reprezentarea finală cerută este o reprezentare generală de cunoştinţe (*KR*) pe care o foloseşte sistemul pentru a reprezenta şi raţiona asupra domeniului său de aplicaţii. Acesta este un limbaj în care sunt reprezentate toate cunoştinţele specifice bazate pe aplicaţii. Scopul unei interpretări contextuale este de a lua o reprezentare a structurii unei propoziţii şi forma sa logică, şi de a o aplica într-o anumită expresie din KR, oferind-o sistemului să o utilizeze în reprezentarea sa.

De obicei se foloseşte ca limbaj de reprezentare finală calculul predicatelor de ordinul I (*FOPC*).

Cursul va urmări trei nivele de reprezentare a limbajului natural: structura sintactică, forma logică şi reprezentarea înţelesului final. Fluxul de informaţie dat de un mesaj trece prin următoarele etape de prelucrare. Această structură vom încerca să o urmărim. Nu vor mai fi dezvoltate elementele formale introduse şi studiate în anii anteriori (limbaje formale, translatori, logică, calcul propoziţional), decât printr-o trecere sumară în revistă în cadrul anexelor. Prelegerile vor completa aceste cunoştinţe cu elemente specifice studierii şi analizei proprietăţilor limbajului natural.

1.6 Libertatea limbajului

Cea ce face deosebit de dificil studiul unui limbaj este dinamica sa, faptul că el se modifică destul de rapid în timp şi spaţiu. Un limbaj natural - spre deosebire de unul artificial - nu poate fi definit strict; de aceea se folosesc anumite convenţii sau norme de prezentare. O parte din ele sunt:

- *Gramatica*: în general regulile gramaticale (privind lexicul) se modifică mult mai greu în timp; de aceea ele pot fi folosite destul de eficient în caracterizarea unei limbi.
- *Norme fonetice şi morfologice.* Sunt structurile de bază folosite în construcţia cuvintelor unei limbi.
- *Regularităţi ale procesului de comunicaţie.* Aici intră convenţii ale actului de vorbire, construcţii specifice folosite în mod mecanic (de genul *Ce mai faci ?, How do you do* etc).

Facem precizarea însă că aceste norme nu sunt obligatorii. Orice vorbitor se poate depărta de ele oricât de mult, atâta timp cât este înţeles (exemplul tipic îl constituie argourile).

Această libertate rezultă direct din natura interactivă a limbajului. Astfel:

- Vorbitorii vor acţiona liber în funcţie de mediu, de variabilitatea situaţiilor sau de contextele verbale;
- Posibilitatea repetiţiilor face ca unele porţiuni de limbaj să fie codificate, prescurtate. Există tendinţa de a nu repeta literar anumite pasaje.

- Aproape orice element de vorbire care poate fi eliminat, va avea în cele din urmă această soartă. Aceasta este generată nu numai din necesitatea de a vorbi mai repede, dar şi din din acea de a înţelege mai uşor semnificaţia mesajului.

Consecinţe ale libertăţii limbajului:

- *Variabilitatea*. Vorbitorul va alege şi va crea colocaţii şi chiar cuvinte noi; alegerea individuală conduce la schimbarea sistemului. De fapt, în acest mod s-a creat fiecare cuvânt din lexic.
- *Posibilitatea de dezvoltare*. Reducerea repetată a morfemelor gramaticale este legată de apariţia unor funcţii ale cuvintelor; acest lucru funcţionează cu condiţia păstrării unui anumit grad de redundanţă a limbii.
- Complexitatea limbilor naturale.
- *Universalitatea*. Nevoia de clarificare a unor înţelesuri noi face necesară dezvoltarea limbii pe toate direcţiile. De remarcat că universalitatea este atât o precondiţie dar şi o consecinţă a libertăţii limbajului. Ea conduce la imposibilitatea definirii de reguli fără excepţii.

Ceea ce face - în ultima instanţă - necesar un studiu sistematic al limbajului natural este observaţia că

"Omul înţelege mai mult decât poate spune dar vorbeşte ca să fie înţeles."

Prelegerea 2

Elemente de lingvistică. Sintaxa limbii

Nu se poate începe un studiu de lingvistică computaţională fără a face o scurtă trecere în revistă a structurii sintactice a limbajului natural. Vor fi tratate în principal elemente de sintaxă ale limbii române, fără a intra în detalii. O dezvoltare de către lingvişti a acestui capitol ar putea conduce la un studiu sistematic aprofundat privind prelucrarea automată a limbii române.

2.1 Cuvinte

La prima vedere, se pare că unitatea de bază în structura unei limbi este *cuvântul*. Totuşi, acesta este departe de a fi elementul fundamental al studiului în lingvistică, el fiind rezultatul aplicării unui set complex de alte elemente (considerate primitive).

Definiţia 2.1 *Disciplina lingvistică numită* **morfologie** *se ocupă cu formarea cuvintelor din diverse componente de bază.*

Există două moduri în care se formează cuvintele noi, moduri numite *forme inflexionale* şi *derivaţionale*.

Formele inflexionale utilizează o componentă numită *rădăcină* a cuvântului, căruia de obicei îi adaugă unul sau mai multe sufixe. Verbele constituie cel mai bun exemplu în această direcţie. Fiecare verb are o formă de bază, care este apoi schimbată în funcţie de persoana şi timpul propoziţiei; se formează astfel conjugările. De exemplu, verbul *a citi* are o rădăcină *cit*, pe baza căreia se formează conjugarea din Tabelul 2.1:

Toate elementele din Tabelul 2.1 sunt verbe şi au acelaşi înţeles.

Morfologia derivaţională foloseşte cuvinte pentru a construi alte cuvinte noi complet diferite (privind înţelesul şi/sau forma gramaticală). De exemplu, substantivul *prieten*, prin adăugarea sufixului *-esc* devine *prietenesc*, care este adjectiv; cu sufixul *-os* se transformă în *prietenos* şi poate fi folosit ca adverb. O formă ceva mai complicată este *împrietenire*. Limba română oferă moduri deosebit de complicate în construirea unor astfel de forme derivaţionale.

Tabelul 2.1:

Timpul prezent	*Timpul perfect simplu*
eu cit-esc	*eu cit-eam*
tu cit-eşti	*tu cit-eai*
el cit-eşte	*el cit-ea*
noi cit-im	*noi cit-eam*
voi cit-iţi	*voi cit-eaţi*
ei cit-esc	*ei cit-eau*

Lingvistica clasifică cuvintele în categorii diferite, în funcţie de utilizarea lor. Se pot considera două zone de interes care contribuie la această împărţire a cuvintelor. Prima zonă se referă la contribuţia cuvintelor în înţelesul frazei din care fac parte, iar a doua zonă este legată de structura sintactică particulară la care participă cuvintele. De exemplu, orice limbaj natural conţine două clase fundamentale:

- Clasa *substantivelor*, formată din acele cuvinte care se folosesc la identificarea tipului de bază al unui obiect, concept sau loc;
- Clasa *adjectivele* - acele cuvinte care atribuie diverse calităţi obiectelor, conceptelor sau locurilor.

Astfel, *verde* este un adjectiv iar *pădure* este un substantiv, după cum se vede şi din construcţia ***pădure verde*** sau ***păduri verzi***. Lucrurile nu sunt însă aşa de simple; ***verde*** poate fi şi substantiv, ca în propoziţia ***Acest verde este mai deschis decât altul***; de asemenea, şi ***pădure*** poate avea alt rol înafară de substantiv - în construcţia ***luminişul pădurii***. Deci, nu există o linie netă trasată între clase. Problema este dacă ele sunt comparabile sau nu. Această afirmaţie este falsă şi o putem observa prin următorul exemplu. Să considerăm toate cuvintele care, completând propoziţia ***Este aşa de ..***, aceasta să rămână semantic corectă. Se poate spune: ***Este aşa de frumos, Este aşa de cald, Este aşa de bine,..*** dar nu putem afirma niciodată ***Este aşa de pădure***.

La fel, nu toate adjectivele pot fi folosite şi ca substantive; aici exemplele care se pot da sunt numeroase.

Fie V alfabetul limbii române şi V^* monoidul liber generat de V cu operaţia de concatenare. Vom nota cu $ST \subset V^*$ clasa substantivelor limbii române, iar cu $AJ \subset V^*$ clasa adjectivelor. Din observaţiile de sus rezultă următoarele relaţii:

$$ST \cap AJ \neq \emptyset, \quad ST \cup AJ \neq ST, \quad ST \cup AJ \neq AJ.$$

Gramatica limbii a pus în evidenţă patru clase importante de cuvinte. Înafară de substantive şi adjective, celelalte două clase sunt: ***verbe*** (V) şi ***adverbe*** (AV). Toate propoziţiile sunt construite folosind ca bază cuvinte din aceste clase. Mai sunt şi alte clase de cuvinte necesare în formarea propoziţiilor, cum ar fi ***articole, pronume, prepoziţii, particule, cuantificatori, conjuncţii***, etc. Dar aceste ultime clase sunt

fixe, în sensul că sunt excepţional de rare cazurile în care apar cuvinte noi care să fie introduse aici. În schimb, substantive, adjective, verbe şi adverbe noi sunt introduse în mod constant odată cu evoluţia limbii. De aceea, aceste clase sunt numite *clase deschise de cuvinte*, în timp ce celelalte sunt *clase închise de cuvinte*.

Orice cuvânt din una din cele patru clase deschise de cuvinte poate fi folosit pentru a forma baza unei expresii. Acest cuvânt va fi numit *liderul* (header) expresiei şi va indica tipul de lucru, activitate sau calitate descris de expresie. Vom avea astfel *expresii-substantiv*, *expresii -adjectiv*, *expresii-verb* şi *expresii-adverb*, fiecare din ele caracterizând o clasă de noţiuni (care pot fi obiecte, însuşiri, acţiuni sau proprietăţi). De exemplu, cu expresiile-substantiv, cuvintele lider indică clasele generale de obiecte descrise. Expresiile

Cabana

Cabana mică

Cabana mică din Bucegi

sunt toate expresii-substantiv care descriu un element din clasa cabanelor. Primul se referă la un element din clasa tuturor cabanelor, al doilea la un element din clasa cabanelor mici, iar al treilea, la un element din clasa cabanelor mici situate în munţii Bucegi. Cuvântul *cabană* este lider în toate aceste expresii.

Uneori, o expresie este formată numai din lider (*nisip* poate fi considerat expresie-substantiv, *slab* poate fi o expresie-adjectiv, *ridicându-se* este expresie-verb, *înfometat* este expresie-adverb). Cel mai adesea însă, liderul cere în expresie cuvinte adiţionale prin care să clarifice înţelesul dorit. De exemplu, verbul *a pune* nu poate forma singur o expresie-verb, înţelesul său nefiind complet. Propoziţia

(*) Ion pune.

este incompletă, ea solicitând continuarea cu o expresie-substantiv, urmată eventual de o expresie exprimând o localizare.

Definiţia 2.2 *Expresia sau setul de expresii necesare pentru a completa înţelesul unui lider se numeşte* **complementul** *liderului.*

Dacă scriem

Ion pune mâna pe carte

Ion este liderul, iar *pune mâna pe carte* este complementul.

În Tabelul 2.2 sunt date exemple de lideri (subliniaţi) din toate cele patru clase majore, însoţite de complemenţi.

2.2 Elementele expresiilor-substantiv simple

Expresiile substantiv (*NP*) sunt folosite pentru a face referire la obiecte, locaţii, concepte, evenimente etc. Cele mai simple astfel de expresii constau dintr-un singur pronume: *el, ea, ei, noi* ş.a.m.d. Pronumele se pot referi la obiecte fizice, ca în propoziţia

El este la uşă.

la evenimente, ca în propoziţia:

După ce am deschis uşa, am regretat-o luni de zile.

sau la calităţi, ca în propoziţia:

Tabelul 2.2:

Expresii-substantiv	Expresii-verb
Preşedintele companiei	*priveşte la trecători*
Dorinţa lui de a reuşi	*credea că lumea este rea*
Diverse provocări ale echipei adverse	*a mâncat o pizza*

Expresii-adjectiv	Expresii-adverb
uşor de trecut	*înăuntrul casei*
fericit că nu a luat premiu	*urgent s-a urcat în tren*
lung ca o girafă	*intermitent priveşte uşa*

Ne era foame, dar nu o arătam.

O altă formă de bază a expresiilor-substantiv constă dintr-un nume sau nume propriu, cum ar fi *Ionescu* sau *Galaţi*. Numele pot fi formate şi din mai multe cuvinte, ca în *Drobeta Turnu Severin* sau *Evenimentul zilei*.

Înafara acestor două cazuri, liderul unei expresii-substantiv este un substantiv comun. De obicei, substantivele se împart în două clase principale:

- **Substantive numărabile**, care descriu obiecte specifice sau mulţimi de obiecte;
- **Substantive amorfe** - referitoare la conţinuturi (compoziţii).

Substantivele numărabile sunt numite astfel deoarece ele pot fi numărate. astfel, pot fi *un câine* sau *mai mulţi câini, o carte* sau *mai multe cărţi* etc. Dacă un substantiv numărabil este folosit pentru a descrie o clasă de obiecte, el va putea fi folosit atât la singular cât şi la plural. Astfel, se poate spune

Câinii sunt prieteni ai omului,

dar şi

Câinele este prieten al omului.

De obicei însă, a doua variantă generează ambiguităţi, necesitând un suport semantic mai puternic (trebuie să ştim că nu se vorbeşte despre un anumit câine); de aceea vom considera ca fiind corectă în specificarea claselor de obiecte, numai forma plurală a substantivelor numărabile.

Substantivele amorfe nu pot fi numărate. Poate fi de exemplu *ceva apă, ceva carne, ceva nisip*; dacă încercăm să introducem un contor, va trebui modificat înţelesul substantivului. De exemplu, *ceva apă* se referă la o cantitate aproximativă de lichid; spunând însă *o apă*, ne gândim la o anumită zonă geografică (un râu, un lac etc). Prin consideraţii similare cu cele de sus, vom folosi pentru descrierea unei clase, numai forma singulară a substantivelor amorfe. Astfel, între

Apa este necesară vieţii.

şi

Apele sunt necesare vieții.

vom alege doar pe prima ca fiind corectă.

Înafară de lider, o expresie poate conține **specificatori** și **calificatori**. Forma generală a unei expresii-substantiv este (cu unele excepții):

{< *specificator* >}{< *calificator* >} < *lider* >

(folosirea acoladelor indică caracterul opțional al conținutului lor).

Calificatorii descriu clasa de obiecte identificată de lider, iar *specificatorii* precizează câte astfel de obiecte sunt sau în ce relație sunt obiectele descrise față de vorbitor sau ascultător.

Specificatorii sunt formați din:

- Numerale ordinale (cum ar fi *primul, al doilea*),
- Numerale cardinale (*unu, doi*,...),
- Articole.

Articolele pot fi împărțite în următoarele clase:

- Hotărât și nehotărât: *un, o, -ul, -a*,...;
- Demonstrativ: *acesta, aceasta, aceia*,...;
- Posesiv: *al (a,ai,ale) lui (lor)*,...;
- Cuantificator: *niște, orice, mult, ambii, jumătate, cel mult, nimic*,...;
- Cuvinte folosite în întrebări: *Cine, ce.*

O expresie-substantiv poate conține cel mult un articol, un numeral ordinal și unul cardinal; poate să aibă din toate trei, cum ar fi: *primii doi admiși.*

Calificatorii constau din adjective și substantive folosite ca atribute. Mai exact:

- **Adjectivele**: sunt cuvinte care acordă anumite proprietăți obiectelor care nu sunt definite prin proprietăți. De exemplu: *vesel, lung, viclean*;
- **Atributele substantivale**: sunt substantive (numărabile sau amorfe) folosite pentru a preciza alt substantiv; de exemplu: *carte de bucate, calculator de buzunar.*

Gramatica asigură substantivelor diverse forme flexionare numite *declinări*. O declinare se face în raport cu numărul (singular, plural), persoana (întâia, a doua, a treia), gen (masculin, feminin, neutru) sau caz (pentru unele tipuri de articole, cum sunt cele posesive).

2.3 Expresii-verb şi propoziţii simple

În vreme ce un NP este folosit pentru a face referire la un obiect, o propoziţie (S) este construită pentru a afirma, a întreba sau a comanda ceva. Se poate afirma că o propoziţie este corectă, se poate întreba dacă o propoziţie este corectă şi se poate cere să se îndeplinească ceea este descris într-o propoziţie. Felul în care este folosită o propoziţie se numeşte *exprimare*. Există patru exprimări de bază ale unei propoziţii, anume:

1. Exprimare **enunţiativă**; Exemplu: *Pisica doarme.*
2. Exprimare **interogativă**; Exemplu: *Cine doarme ?*
3. Exprimare **imperativă**; Exemplu: *Ia pisica asta de aici !*
4. Exprimare **interogativă**, cu răspuns **da/nu**; Exemplu: *Doarme pisica ?*

Ultima exprimare nu este specifică gramaticii limbii române. Totuşi, în această lucrare, vom introduce anumite diferenţe faţă de definiţiile lingvistice, necesare în prelucrarea matematică a textelor.

Definiţia 2.3 *O propoziţie enunţiativă simplă este formată dintr-un NP - cu rol de subiect - urmat de o expresie verbală (VP) folosită ca predicat.*

O VP simplă este de forma

$$\{< modificator - adverb >\} < verb > \{< complemente >\}$$

Verbul apare într-o formă dată de aşa numita *conjugare* a sa, definită în funcţie de patru componente: *mod, timp, persoană* şi *număr*. De exemplu, conjugarea verbului ***a lucra*** la modul indicativ şi timpul prezent este:

Persoana I singular:	*Eu lucrez.*
Persoana II singular:	*Tu lucrezi.*
Persoana III singular:	*El (Ea) lucrează.*
Persoana I plural:	*Noi lucrăm.*
Persoana II plural:	*Voi lucraţi.*
Persoana III plural:	*Ei (Ele) lucrează.*

Nu vom intra în detalii lingvistice referitoare la modul de formare al conjugărilor; pentru limba română, ele sunt deosebit de dificile.

În funcţie de verbul din structura VP se pot deduce multe tipuri de complemente. Astfel, multe verbe pot fi folosite fără nici un complement; ele sunt numite ***verbe intranzitive***. De exemplu, ***a râde*** *(Sergiu râde)*, ***a alerga*** *(Câinele aleargă)*.

Alte verbe trebuie să aibă drept complement expresii-substantiv; acestea sunt ***verbele tranzitive***, cum ar fi de exemplu ***a găsi*** *(Pisica găseşte drumul către casă)*, ***a duce***, ***a trage***.

Cele două clase de verbe sunt incomparabile; există verbe care aparţin unei singure clase, şi verbe care pot fi - în funcţie de context - atât tranzitive cât şi

intranzitive. De exemplu, *a cânta* poate fi tranzitiv sau intranzitiv, dar înţelesul diferă; astfel, se poate spune *Radu cântă* dar şi *Radu cântă o melodie cunoscută.*

Dacă verbele intranzitive sunt caracteristice diatezei active, cu verbele tranzitive se poate forma diateza pasivă, construită din verbul *a fi* şi participiul trecut al verbului. În diateza pasivă, expresia-substantiv care este scrisă pe poziţia complementului va juca rol de subiect. Exemple referitoare la această trecere sunt date în Tabelul 2.3.

Tabelul 2.3:

Diateza activă	Diateza pasivă
George aruncă mingea.	*Mingea este aruncată de George.*
Profesorul l-a lăudat.	*A fost lăudat de profesor.*
Voi vizita oraşul.	*Oraşul va fi vizitat de mine.*

Câteva observaţii:

- Elementele legate de verb (timp, mod, etc.) se păstrează în trecerea de la o diateză la alta.
- Deşi prima expresie-substantiv pare că a trecut în forma pasivă pe poziţia complementului, ea a rămas din punct de vedere semantic subiectul.

Unele verbe permit două expresii-substantiv pe post de complement. De exemplu:
Petru îi va da Alinei o carte.
Radu a primit într-o oră informaţiile cerute.
În această situaţie, primul NP este numit *NP direct*, iar al doilea - *NP indirect.*

În general, astfel de propoziţii pot fi transformate în altele echivalente în care expresia-substantiv indirectă este separată prin construcţii de tip prepoziţie. Astfel, se poate spune:
Petru îi va da o carte lui Alina.
Radu a primit într-o oră informaţiile pe care le-a cerut.

2.4 Complemente directe

Multe verbe solicită să fie însoţite de complemente (să le numim *clauze*); acestea au diverse forme, de la simple particule până la propoziţii complete (completive directe, indirecte, etc). Uzual, ele sunt precedate de anumite particule, cum ar fi: <u>*că*</u> *(Aud că eşti acasă)*. O astfel de clauză va fi identificată prin expresia **S[că]**, indicând o subclasă specială a structurii S. Aceasta poate apare ca un complement direct al verbului *a şti*, sub toate diatezele:

1. activă: (*Eu ştiu că Petre a mâncat pizza*);
2. pasivă: (*Eu ştiu că pizza a fost mâncată de Petre*);

3. reflexivă:(*Eu ştiu că pizza se mănâncă caldă*).

Alt tip de clauze foloseşte forma infinitivă a verbelor: **VP[inf]** este un VP care începe cu infinitivul. Exemplul tipic este versul eminescian

Nu credeam să învăţ a muri vreodată.

S[inf] poate apare şi prin folosirea particulei *pentru*, ca în

Ion s-a închis în casă pentru a avea linişte.

În mod similar se pot defini diverse complemente directe semnalate şi de alte particule: *ce, cine, unde, câte* etc.

Există verbe care - cu foarte rare excepţii - nu pot fi folosite decât însoţite de expresii-complement; mulţimea prepoziţiilor folosite pentru introducerea acestor complemente va fi notată PP.

Exemplul 2.1 *Verbul "a crede" foloseşte drept PP prepoziţia "că":*

Eu cred că veşnicia s-a născut la sat.

"a pune" foloseşte un complement format dintr-un NP precedat de un PP care descrie o locaţie:

Ion pune cartea pe masă.

Ion pune cartea în raft.

Ion pune cartea lângă celelalte.

Spunem că "a pune" are un complement de forma **NP+PP[loc]**.

Similar, "a decide" poate avea un complement de forma **NP+PP[despre]**, *"a lucra"* - **NP+PP[la]**, *ş.a.m.d.*

În Tabelul 2.4 sunt strânse câteva structuri de complemenţe utilizate în limba română. Toate variantele ar însuma peste 50 forme şi nu am cunoştinţă de existenţa unei liste complete de astfel de structuri.

2.5 Expresii-substantiv complete

Folosirea unei expresii-substantiv formată numai dintr-un element - lider (aşa cum a fost introdus anterior) este incompletă. Ea poate avea forme mai complexe, în care un NP poate conţine propoziţii întregi sau expresii-verb drept subcomponente.

Toate exemplele anterioare aveau lideri fără complemente; totuşi, multe expresii-substantiv solicită expres complemente cu anumite expresii- prepoziţie. De exemplu, substantivul *dragoste* are un complement de forma **PP[pentru]**, ca în *dragostea ei pentru Franţa*; *legătură* poate avea un complement de forma **PP[cu]**, ca în *legătura lor cu exteriorul*, etc.

Multe substantive, cum ar fi *dorinţă, nerăbdare, curiozitate* ... pot avea completare un VP la infinitiv, ca în *dorinţa lui de a pleca, nerăbdarea de a vedea ceva nou, curiozitatea de a cerceta*. Ele admit de fapt forma **S[inf]** drept complement.

Expresiile substantiv pot fi construite şi pe clauze, aşa cum au fost introduse în paragraful precedent. De exemplu, o clauză *că* (**S[că]**) poate fi folosită ca subiect al unei fraze, cum ar fi:

Că George este însurat, nu a mirat pe nimeni.

De asemenea pot fi folosite drept expresii-substantiv:

Tabelul 2.4:

Verb	Structura complementului	Exemplu
a râde	∅ (intranzitiv)	Radu râde.
a găsi	NP (tranzitiv)	Radu a găsit o cheie
a da	$NP + NP$ (bitranzitiv)	Poştaşul a dat plicul destinatarului.
a da	$NP + PP$[la]	Poştaşul a dat scrisoarea la retururi.
a sta	Completivă de loc	Radu sta la ţară.
a pune	NP+ Completivă de loc	Radu a pus haina în cuier.
a vorbi	PP[cu] + PP[despre]	Radu vorbeşte cu Ion despre mine.
a încerca	VP[să]	Radu încearcă să vorbească.
a spune	$NP + VP$[să]	Radu iăspus musafirului să intre.
a dori	S[să]	Radu doreşte ca totul să fie perfect.
a râde	VP[âng]	Radu râde plângând.
a chema	$NP + VP$[âng]	Radu chema câinele fluierând.
a regreta	S[că]	Radu regretă că a pierdut trenul.
a spune	$NP + S$[că]	Radu spune fiului să plece la şcoală.
a părea	$ADJP$	Radu pare nefericit la serviciu.
a şti	S[unde]	Radu ştie unde sunt fraţii lui.

1. Forme infinitive ale expresiilor-verb: **VP[inf], S[inf]**. De exemplu:

   A câştiga o maşină este o problemă de şansă.

   Pentru Andrei, a ajunge la timp este ceva neobişnuit.

2. Forme gerundive: **VP[âng], S[âng]**. Exemple:

   Terminând treaba, am plecat acasă.

   Aflând că Ion este plecat, nu l-am mai sunat.

2.6 Expresii - adjectiv

După cum am văzut - cele mai simple expresii-adjectiv ($ADJP$) constau dintr-un simplu cuvânt cu rol de adjectiv. Sunt posibile însă şi expresii-adjectiv mai complexe, adjectivele putând fi însoţite de aceleaşi forme complement tratate la expresiile-verb. Astfel,

- Se pot folosi expresii prepoziţionale specifice: *cu*, *la*, ca în

  Lui Ion i-a plăcut cu trenul.

  Anca a fost supărată la plimbare.

- Se pot folosi forme complement cum ar fi **S[că]**:

  Paul s-a supărat că n-a putut intra la meci.

- Sunt adjective care se pot construi din forme infinitive sau conjunctive ale verbelor (**VP[inf], VP[conj]**):

Nu am învăţat a mânui condeiul.

Anca doreşte să ajungă arhitect.

Astfel de construcţii complexe de expresii-adjectiv sunt specifice pe poziţii de complement ale anumitor verbe (*a fi, a părea*) sau după liderul unei expresii substantiv.

2.7 Expresii-adverb

Expresiile-adverb pot apare în diverse poziţii în propoziţii:
- pe prima poziţie, ca în:
  Pe urmă Anca a deschis radioul.
- în expresii-verb:
  Anca a deschis pe urmă radioul.
- pe ultima poziţie a propoziţiei:
  Anca a deschis radioul pe urmă.

Există bineînţeles restricţii ale formelor de adverb care pot fi folosite în fiecare poziţie, dar acestea sunt legate strâns de structura internă a fiecărei limbi.

Totuşi, din cauza numărului mare de forme, în general este util ca expresiile-adverb (*ADVP*) să fie considerate după funcţii, nu după forma sintactică. Deci, putem avea expresii-adverb temporale, de durată, de locaţie, grad, frecvenţă, mod etc, fiecare în forma sa specifică.

De exemplu, expresiile-adverb temporale se pot folosi ca particule adverbiale (*acum*), expresii-substantiv (*astăzi, ieri*), expresii-prepoziţionale (*la prânz, în timpul filmului*) şi clauze (*când ceasul a sunat de amiază, înainte ca lupta să înceapă*).

În această lucrare, adverbele vor apare adesea ca modificatori ai acţiunilor sau stărilor descrise de propoziţie. De obicei, asemenea situaţii ridică problema distingerii între complementele verbului şi aceste adverbe. O distincţie uzuală este aceea că expresiile adverbiale sunt întotdeauna opţionale. Deci, prin eliminarea lor se obţine o propoziţie cu aproximativ acelaşi înţeles. Să considerăm ca exemplu propoziţiile:

Ion taie picioarele la masă.
Ion mănâncă o prăjitură la masă.

Expresia *la masă* este un complement în prima propoziţie (eliminarea ei duce la o propoziţie "horror", fără înţeles) şi o expresie adverb în a doua (important este că *Ion* mănâncă o prăjitură, locul unde se petrece această acţiune fiind o informaţie suplimentară).

Prelegerea 3

Noţiuni de morfologie

3.1 Structura morfologică a cuvintelor

Regulile de formare ale cuvintelor - care pornesc de la o anumită structură a acestora - diferă de la o limbă la alta, atât prin numărul lor cât şi prin complexitate. Aceste reguli permit uneori ca pe baza unei rădăcini să se formeze un număr impresionant de cuvinte înrudite. Limbile aglutinate (germana, maghiara, finlandeza) folosesc mult acest procedeu. În limba finlandeză de exemplu, pornind de la rădăcina unui substantiv se pot forma prin procedee morfologice câteva mii de cuvinte valide şi interpretabile de către un vorbitor nativ. Pentru verbe se pot genera uneori peste 10.000 de forme diferite ale aceluiaşi cuvânt. Fenomenul este similar - la o scară mai redusă - şi pentru limbile neaglutinate. De pildă, în limba rusă prin conjugarea unui verb se pot obţine în jur de o sută de forme, cifră aproximativ valabilă şi pentru limba română (dacă luăm în considerare şi formele compuse). Deci cuvintele evidenţiază o anumită structură ce furnizează o mare bogăţie informaţională care nu poate fi ignorată.

În structura unui cuvânt se poate distinge o parte constantă şi una variabilă. Statutul de *parte constantă* într-un cuvânt nu este absolut, ci relativ la o anumită familie de derivaţi morfologici.

Definiţia 3.1 *([5]) Orice cuvânt este format dintr-o* **rădăcină** *la care se adaugă unul sau mai multe* **afixe**. *Afixele pot fi* **prefixe, sufixe** *şi* **desinenţe**.

Rădăcina este elementul neanalizabil din punct de vedere morfologic, fiind comună mai multor cuvinte cu sens înrudit (deşi pot aparţine unor părţi de vorbire diferite).

Prefixul este afixul adăugat înaintea rădăcinii. În limba română prefixele au numai valoare lexicală, de elemente cu care se formează cuvinte noi.

Sufixele sunt afixe adăugate la sfârşitul rădăcinii; ele sunt de două feluri: *sufixe lexicale* (sufixe cu valoare lexicală care conduc la formarea de cuvinte noi) şi *sufixe gramaticale* (cu valoare gramaticală, care ajută la formarea de forme flexionare). Dacă o rădăcină are mai multe sufixe, primele care se scriu sunt sufixele lexicale, apoi cele gramaticale.

Rădăcina împreună cu prefixele şi sufixele formează **tema**. Spre deosebire de rădăcină, tema este comună numai formelor unuia şi aceluiaşi cuvânt. De exemplu,

în familia cuvântului *parte* există rădăcina **part** din care se pot crea o mulţime de teme: *părtaş-*, *împărţi*, *împărtăş-* etc, caracteristice fiecărui cuvânt.

Desinenţele sunt sufixe gramaticale care se adaugă după rădăcină sau temă pentru a arăta:

- numărul şi cazul (la substantive);
- genul, numărul şi cazul (la adjective);
- persoana şi numărul (la verbe).

Spre deosebire de prefixe şi sufixe care pot intra mai multe în compoziţia unui cuvânt, o formă gramaticală nu poate conţine decât o singură desinenţă.

3.2 Prelucrarea morfologică a cuvintelor

După cum am specificat în prelegerea precedentă, lingvistica computaţională cuprinde actualmente două maniere diametral opuse privind abordarea problemelor legate de structura cuvintelor; ele sunt **paradigma procedurală** cu formele sale **derivativă** şi **flexionară**, respectiv **paradigma declarativă**.

Poziţia adoptată faţă de analiza cuvintelor determină în mare parte organizarea şi conţinutul dicţionarului, care este elementul central al prelucrării la nivel morfologic.

Dicţionarul este recunoscut unanim ca fiind o componentă esenţială a oricărui sistem de prelucrare a limbajului natural. În contrast cu limbajele de programare care necesită dicţionare mici, conţinând elemente lexicale neambigue, limbajele naturale se sprijină pe volume lexicale incomparabil mai mari, elementele de dicţionar fiind rareori univoce din punct de vedere morfologic, sintactic şi mai ales semantic.

Problemele apar chiar din momentul definirii conţinutului unui dicţionar; ele sunt generate de diversitatea orientărilor teoretice şi practice, de specificitatea fiecărei limbi naturale, de gradul de detaliu şi profunzime la care se ajunge în prelucrarea limbajului, de investiţia de muncă şi de resursele hard şi soft aflate la dispoziţie.

Definiţia 3.2 *Conform paradigmei derivative, dicţionarul cuprinde doar rădăcinile cuvintelor împreună cu toate afixele caracteristice (lexicale şi gramaticale).*

Deci, în acest caz, cuvintele limbii sunt recunoscute sau generate în urma unui proces derivativ.

Obiecţii:

- Complexitatea mare a proceselor derivative, de regulă nedeterministe, ceea ce conduce în multe cazuri la probleme NP-complete.
- Dificultatea specificării algoritmice a unui set complet de reguli derivative capabile să acopere fenomenul dinamic al formării cuvintelor şi a prelevării rădăcinilor şi afixelor semnificative.
- Existenţa (în multe limbi) de false afixe, ceea ce îngreunează procesul de recunoaştere a rădăcinilor şi chiar a temelor.

- Din punct de vedere semantic, este greu de decis algoritmic formarea şi modificarea semnificaţiei cuvintelor formate derivativ. De exemplu, este greu de derivat semnificaţia cuvântului *împărtăşanie* plecând de la rădăcina *-part-*.

- Folosind regulile de derivare se pot obţine cuvinte inexistente în fondul lexical al unei limbi.

Cu toate acestea, avantajul major al unei abordări derivative constă în posibilitatea de reducere a dicţionarelor (de rădăcini) la dimensiuni modeste, necesitând un efort constructiv rezonabil. Pe de-altă parte, chiar unele obiecţii pot conduce la avantaje. De exemplu, dinamica unei limbi solicită frecvent introducerea de cuvinte noi; acestea se supun regulilor flexionare şi derivative deja existente, reguli care se modifică mult mai greu (în timp).

Definiţia 3.3 *Un dicţionar construit conform paradigmei declarative va conţine (explicit sau nu) toate informaţiile structurale ale cuvintelor unei limbi.*

Deci, într-o abordare declarativă, un dicţionar nu va conţine rădăcini sau teme, ci cuvinte; de exemplu, odată cu cuvântul *program*, în dicţionar se vor găsi şi *programe, programele, programul, programului, programelor* etc. Informaţiile contextuale legate de forma flexionară a unui cuvânt sunt reprezentate explicit. Astfel, pentru intrarea corespunzătoare cuvântului *programelor*, dicţionarul va conţine explicit informaţii de tipul:

- **forma-normală**: program;
- **numărul**: plural;
- **cazul**: genitiv/dativ
- **articulare**: enclitic
- ...

Aceasta este ideea de bază în construcţia corpusului unei limbi.

Procesul morfo - lexical se va reduce aici la simpla căutare a elementului curent în dicţionar, eliminându-se complet nedeterminismul întâlnit la abordarea procedurală. Cuvintele înrudite sunt de obicei puse în corespondenţă prin aşa numitele ***reguli de redondanţă***.

Doarece nu ne vom ocupa de această manieră de construcţie, nu vom detalia mai departe argumentele pro şi contra utilizării procedurii declarative.

Analiza morfo-lexicală constă din câteva subfaze specifice, care apar explicit în abordările procedurale. Acestea sunt:

1. Primul obiectiv este prelucrarea individuală a cuvintelor, ceea ce înseamnă identificarea rădăcinii sau temei, precum şi a afixelor modificatoare. Structura rezultată este numită de obicei *atom morfologic*.

2. Atomii morfologici sunt supuşi unui proces reductiv, numit ***analiza perifrastică***, care are rolul de a identifica formele compuse ale verbelor, gradele de comparaţie ale adjectivelor şi adverbelor precum şi mijloacele analitice de flexionare (construcţii cu propoziţii, conjuncţii, articole). Ceea ce se obţine este o secvenţă de structuri de date numite *atomi morfo - lexicali*.

3. Atomii morfo - lexicali sunt prelucraţi pentru a identifica expresiile compuse cu semnificaţie lexicală stabilă (cum ar fi *locuţiunile* şi *sintagmele*); această prelucrare este numită **analiză sintagmatică**. Rezultatul final este o listă de atomi lexicali care constituie baza prelucrărilor ulterioare.

Structura atomilor morfologici, morfo - lexicali şi lexicali depinde de mai mulţi factori care se influenţează reciproc, anume:

- Teoria lingvistică (sintactică, semantică, pragmatică) pe care se clădeşte sistemul de prelucrare al limbajului natural;
- Situaţia conversaţională şi universul de discurs modelate;
- Tipul aplicaţiei proiectate.

În raport cu aceste condiţionări, structura şi conţinutul dicţionarului pot varia foarte mult. Un element de dicţionar poate conţine, înafara rădăcinii, temei sau cuvântului propriu-zis (după cum se foloseşte paradigma procedurală respectiv declarativă):

- Un câmp etimologic (mai ales la abordările derivative);
- Un câmp fonologic, descriind structura proozodică (la dicţionarele destinate limbajului vorbit sau la editoarele de texte, unde se foloseşte de exemplu la despărţirea în silabe);
- Un câmp sintactic;
- Un câmp semantic, descriind semnificaţiile lexicale asociate rădăcinii, temei sau cuvântului, relaţiile de sinonimie, antonimie, polisemie.
- Un câmp pragmatic, sepcificând construcţii uzuale, interpretări preferenţiale sau probabile, modificări ale semnificaţiilor lexicale generale, etc.

3.3 Morfologia pe două nivele

Cea mai cunoscută teorie din zona paradigmei derivative este morfologia pe două nivele, prezentată prima oară de Kimmo Koskenniemi în teza sa de doctorat din 1983. Ea s-a impus mai ales datorită fundamentării teoretice solide şi uşurinţei de implementare.

În esenţă, modelul lui Koskenniemi foloseşte două nivele în reprezentarea cuvintelor:

1. *Nivelul de suprafaţă*, corespunzător apariţiei cuvintelor în text;

2. *Nivelul lexical*, corespunzător ortografierii cuvintelor în dicţionar.

Esenţa teoriei constă în utilizarea unor reguli care încearcă să generalizeze fenomenele morfologice prin corelarea reprezentării nivelului lexical cu reprezentarea de suprafaţă.

Definiţia 3.4 *Fiind date două alfabete finite nevide* Σ *şi* Γ*, se numeşte* **pereche - simbolică** *un element* $(a,b) \in \Sigma \times \Gamma$.

În notaţia consacrată, perechea simbolică (a,b) este reprezentată de $a : b$ în care caracterul : este un metasimbol al modelului.

Se numeşte **secvenţă de perechi simbolice** o succesiune (posibil vidă)

$$a_1 : b_1, a_2 : b_2, \ldots, a_n : b_n$$

de perechi simbolice.
O mulţime $L \subseteq (\Sigma \times \Gamma)^*$ se numeşte **limbaj de perechi simbolice** peste Σ şi Γ.

Fiind date alfabetele Σ şi Γ şi o secvenţă S de perechi simbolice, o secvenţă $< P_1, \ldots, P_n >$ de perechi simbolice se numeşte **partiţie** a lui S dacă şi numai dacă $S = P_1 P_2 \ldots P_n$.

Definiţia 3.5 *O* **regulă morfologică** *pe două nivele peste alfabetele* Σ *şi* Γ *este un cuplu* $< P, C >$ *unde* P *este o pereche simbolică iar* C *este o mulţime nevidă de perechi* $< LC, RC >$ *unde* LC *şi* RC *sunt expresii regulate peste* $\Sigma \times \Gamma$ *numite* **contexte stângi** *respectiv* **drepte**.

O expresie context EC *satisface la dreapta (stânga)* o secvenţă S de perchi simbolice dacă şi numai dacă există o partiţie $< P_1, P_2 >$ a lui S astfel încât P_2 (respectiv P_1) aparţine mulţimii descrise de EC.

3.3.1 Reguli morfologice cu două nivele

Există două tipuri de reguli morfologice de bază pe două nivele; anume:

1. **Regula de restricţionare contextuală (RRC).**

   O regulă RRC de forma

   $$I : i \longrightarrow b : b - c : c$$

   semnifică faptul că perechea simbolică $I : i$ este legală (corectă) doar în contextul perechilor simbolice $b : b$ şi $c : c$ care obligatoriu preced respectiv succed perechea simbolică $I : i$.

   Ca o observaţie, s-a folosit $\longrightarrow$ ca separator al celor două componente $< P, C >$ din definiţia formală a unei reguli morfologice pe două nivele.

2. **Regula de restricţionare a formei de suprafaţă (RRS).**

   O regulă RRS de forma

   $$I : i \longleftarrow b : b - c : c$$

specifică faptul că ori de câte ori apar contextele respective ($b:b$ la stânga şi $c:c$ la dreapta) şi se cunoaşte simbolul din Σ (l în formulă), atunci el este obligatoriu asociat cu simbolul specificat din Γ (i în formulă).

Ca o observaţie, s-a folosit $\longleftarrow$ ca separator al celor două componente $< P, C >$ din definiţia formală a unei reguli morfologice pe două nivele.

Înafara acestor două reguli, se mai foloseşte şi un al treilea tip de regulă care are ca separator $\longleftrightarrow$. O regulă $P \longleftrightarrow Q$ este echivalentă cu pereche de reguli $P \longrightarrow C$ şi $P \longleftarrow C$.

Să comentăm semnificaţia simbolurilor folosite în regulile morfologice cu două nivele:

Alfabetul Σ se numeşte *alfabetul lexical* şi conţine:

- Coduri pentru literele alfabetului limbii naturale a cărei morfologie se descrie;
- Simbolul + pentru separarea morfemelor;
- Metasimboluri (notate cu litere latine mari) pentru notarea submulţimilor lui Σ. De exemplu, **C** poate defini mulţimea consoanelor din Σ, iar **V** - mulţimea vocalelor.

Alfabetul Γ se numeşte *alfabetul de suprafaţă* şi conţine:

- Literele şi diacriticele ce pot apare în grafia unui cuvânt;
- Metasimboluri pentru submulţimi din Γ;
- Simbolul vid $\emptyset$.

Exemplul 3.1 *Pentru ilustrarea notaţiei folosite în modelul lui Koskenniemi, să considerăm regula următoare:*

$$\begin{array}{ll} e:\emptyset \longleftrightarrow & \{=: C_2- < +:\emptyset\ V := >\} \\ & \{< C:C\ V:V > - < +:\emptyset\ e:e >\} \\ & \{c:c- < +:\emptyset\ a:\emptyset\ t:t >\} \end{array}$$

În această regulă, $= \subset \Sigma \cup \Gamma, V \subset \Sigma, C \subset \Sigma \cup \Gamma, C_2 \subset \Sigma, V \subset \Gamma$. *Perechile simbolice închise între paranteze unghiulare reprezintă secvenţe, iar perechile simbolice cuprinse între acolade reprezintă alternative.*

Deci perechea simbolică $e:\emptyset$ *este legală numai dacă apare în contextele listate în partea dreaptă a regulii, în poziţia indicată de simbolul* $-$. *Fiecare context are o parte stângă şi o parte dreaptă, care sunt fie perechi simbolice fie expresii regulate cu perechi simbolice.*

Interpretarea regulii (vom explicita doar componenta $\longrightarrow$ *a relaţiei* $\longleftrightarrow$*) este următoarea: transcrierea unui element lexical (morfem, lemă, rădăcină) într-unul de suprafaţă se face prin înlocuirea simbolului lexical* e *cu caracterul de suprafaţă* $\emptyset$ *(deci se şterge litera e în procesul de generare a formei grafemice) dacă apare una din variantele următoare:*

a. *Simbolul lexical e este precedat în reprezentarea lexicală de un simbol din clasa = care în transcrierea de suprafaţă aparţine clasei C_2 şi este urmat de un delimitator morfemic (+) care se va şterge în reprezentarea de suprafaţă; după acest delimitator urmează un simbol lexical din clasa V care se rescrie în forma de suprafaţă printr-un simbol din clasa =;*

b. *Simbolul lexical e este precedat în reprezentarea lexicală de două simboluri din clasele C respectiv V, care se vor rescrie în simbolurile de suprafaţă aparţinând aceloraşi clase; el este urmat de un delimitator morfemic (care se va şterge) şi simbolul lexical e care rămâne nemodificat în reprezentarea de suprafaţă;*

c. *Simbolul lexical e este precedat în reprezentarea lexicală de simbolul c (scris în reprezentarea de suprafaţă); el este urmat de de secvenţa lexicală formată de delimitatorul morfemic, simbolul a (şters în reprezentarea de suprafaţă) şi simbolul t (nemodificat în reprezentarea de suprafaţă).*

Vom mai da un exemplu - practic - de folosire a acestui model.

Exemplul 3.2 *Regula de geminaţie consonantică (dublarea consoanei finale a unui morfem) în limba engeză este:*

$$+ : \mathbf{CG} \longleftrightarrow =: \mathbf{CG} - \mathbf{V} : \mathbf{V}$$

unde

$$\mathbf{CG} \in \{\mathbf{b},\mathbf{d},\mathbf{f},\mathbf{g},\mathbf{l},\mathbf{m},\mathbf{n},\mathbf{p},\mathbf{r},\mathbf{s},\mathbf{t}\}, \mathbf{V} \in \{\mathbf{a},\mathbf{e},\mathbf{i},\mathbf{u}\}.$$

Să urmărim cum funcţionează această regulă, atât în generarea cât şi analiza unui cuvânt:

Fie morfemul "**refer**" *aparţinând dicţionarului. Generarea gerunziului (sau trecutului) presupune selectarea morfemului* **+ing** *(sau* **+ed***) astfel încât prin concatenare se obţine şirul lexical* **refer+ing** *(sau* **refer+ed***) deoarece delimitatorul morfemic + apare în contextul cerut de regula precedentă:* $\mathbf{r} \in \mathbf{CG}, \mathbf{i(e)} \in \mathbf{V}$, *în reprezentarea de suprafaţă simbolul din* **CG** *substituind simbolul +. Se obţine astfel forma* **referring** *(sau* **reffered***).*

În analiza cuvântului **referring** *(sau* **referred***), după separarea sufixului* **ing** *(respectiv* **ed***), este necesară verificarea faptului că în cuvântul analizat acest subşir este chiar morfemul* **+ing** *(de exemplu, în cuvinte ca "sting", "king", el nu este un morfem). Aceasta presupune ca în reprezentarea lexicală obţinută prin partiţionarea implicită a cuvântului analizat*

$$< \mathbf{r}:\mathbf{r}\ \mathbf{e}:\mathbf{e}\ \mathbf{f}:\mathbf{f}\ \mathbf{e}:\mathbf{e}\ \mathbf{r}:\mathbf{r} > + : \mathbf{r} < \mathbf{i}:\mathbf{i}\ \mathbf{n}:\mathbf{n}\ \mathbf{g}:\mathbf{g} >$$

simbolurile lexicale se reprezintă în morfeme existente în dicţionar. Deoarece **refer** *şi* **+ing** *sunt morfeme lexicale, iar regula de geminare consonantică permite rescrierea unui simbol de suprafaţă din* **CG** *în +, analiza cuvântului* **referring** *ca suma morfemelor "refer" respectiv "+ing" este corectă.*

3.3.2 Gramatica morfologică pe două nivele

Definiţia 3.6 *Un set de reguli morfologice pe două nivele* **permite contextual** *o secvenţă S de perechi simbolice dacă şi numai dacă pentru fiecare partiţie*

$$< P_1,\ a:b,\ P_2 >$$

a lui S este valabilă una din următoarele afirmaţii:

- *Nu există în* **R** *nici o regulă de forma* $< a:b, C >$*;*
- *În* **R** *există cel mult o regulă de această formă, cu* **C** *conţinând o pereche de contexte* $< LC, RC >$ *astfel încât* $P_1 \in LC, P_2 \in RC$.

Definiţia 3.7 *O regulă morfologică pe două nivele* $<< a : b >,\ C >$ **permite coercitiv** *o secvenţă* **S** *de perechi simbolice dacă şi numai dacă pentru orice partiţionare posibilă a lui* **S** *în* $< P_1,\ a':b',\ P_2 >$ *şi pentru fiecare element* $< LC, RC > \in C$ *cu* $P_1 \in LC, P_2 \in RC$, *dacă* $a = a'$ *atunci* $b = b'$.

O gramatică morfologică cu două nivele pentru alfabetele Σ, Γ se defineşte formal printr-o pereche $< RRC, RRS >$ unde **RRC** este o mulţime de reguli de restricţionare contextuală peste Σ şi Γ, iar **RRS** este o mulţime de reguli de restricţionare a formelor de suprafaţă peste Σ şi Γ.

O pereche simbolică $a : b$ este **fezabilă** într-o gramatică cu două nivele definită peste alfabetele Σ, Γ dacă şi numai dacă:

- $a = b$ şi $a \in \Sigma \cup \Gamma$, sau
- $a \in \Sigma, \beta \in \Gamma$ şi $a : b$ apare într-o regulă din gramatică.

O pereche $a : b$ apare într-o regulă $<< a' : b >, C >$ dacă:

- $a = a',\ b = b'$ sau
- $\exists < LC, RC > \in C$ astfel încât $a : b \in LC \cup RC$.

Definiţia 3.8 *Fiind dată o gramatică morfologică cu două nivele* $G =< RRC, RRS >$, *o secvenţă S de perechi simbolice este* **generată** *de G dacă şi numai dacă:*

1. *Toate perechile simbolice din S sunt fezabile;*
2. *Orice regulă din RRS permite coercitiv S;*
3. *Mulţimea regulilor din RRC permite contextual S.*

Observaţii:

1. Conform definiţiei de mai sus, regulile din *RRS* formează o mulţime conjunctivă de restricţii care trebuiesc satisfăcute simultan, în timp ce regulile din *RRC* formează o mulţime disjunctivă de restrciţii specificând toate contextele legale posibile.

2. Din definiţia mulţimilor *RRC* şi *RRS* rezultă că dacă nici o regulă nu se aplică unei perechi simbolice, ea este acceptaă dacă şi numai dacă este fezabilă.

Definiţia 3.9 *Fiind date* $\alpha = s_1 s_2 \dots s_n \in \Gamma^*, \beta = I_1 I_2 \dots I_n \in \Sigma^*$ *şi o gramatică morfologică cu două nivele G, spunem că* β *este* **reprezentarea lexicală a lui** α **în** *G dacă secvenţa de perechi simbolice* $I_1 : s_1\ I_2 : s_2 \dots I_n : s_n$ *este generată de G.*

Din definiţia de sus se observă restricţia foarte puternică $|\alpha| = |\beta|$; ea poate fi relaxată permiţând ca $I_i \in \Sigma \cup \{\emptyset\}, s_i \in \Gamma \cup \{\emptyset\}$.

Procesul de prelucrare morfologică în morfologia pe două nivele poate fi abstractizat folosind două noţiuni suplimentare: *sistemul de segmentare lexicală* şi *dicţionarul.*

Definiţia 3.10 *Un sistem de segmentare lexicală într-o morfologie cu două nivele este o structură* $(\Sigma, \Gamma, \emptyset, L_x, f, G)$ *unde:*

- Σ *şi* Γ *sunt alfabetele lexical respectiv de suprafaţă;*
- $\emptyset \notin \Sigma \cup \Gamma$ *este un simbol special;*
- $L_x = \{L_1, \dots, L_n\}$ *este o mulţime de mulţimi de cuvinte din* Σ^* *numită lexicon, fiecare* L_i *fiind un sublexicon;*
- $f : \Sigma^* \longrightarrow 2^L$ *este o funcţie definită astfel:* $\forall w \in L_i, f(w) = \{L_{i_1}, \dots, L_{i_k}\} \subseteq L_x$. $f(w)$ *se numeşte mulţimea de continuare validă a morfemului lexical w;*
- *G este o gramatică morfologică cu două nivele.*

Cu această definiţie, procesul de segmentare validă a unui cuvânt în elementele sale lexicale constitutive se descrie natural astfel:

Definiţia 3.11 *Fiind date cuvântul* $\alpha \in \Gamma^*$ *şi* β *reprezentarea sa lexicală,* α *este segmentat în* $< I_1, \dots, I_n >$ *dacă:*

- $I_1 \dots I_n = \beta'$ *unde* β' *este şirul* β *din care s-au eliminat eventualele morfeme nule;*
- $\exists L_i$ *astfel încât* $I_i \in L_i, L_i \in L_x, 1 \leq i \leq n;$
- $L_{i+1} \in f(I_i), 1 \leq i \leq n-1.$

Pentru ca această schemă să fie operaţională, dicţionarul - în modelul lui Koskenniemi - este structurat în felul următor: fiecare intrare lexicală (corespunzătoare unui morfem) conţine - pe lângă informaţiile de natură morfologică şi lexicală proprii morfemului respectiv - una sau mai multe clase de continuare. Acestea definesc tipurile de morfeme care pot fi concatenate la dreapta cu morfemul curent în structura unui cuvânt valid. Morfemele nu sunt înregistrate într-un singur dicţionar, ci sunt grupate în sublexicoane, fieare din acestea corepunzând unei singure clase morfo-sintactice.

În general clasele de continuare induc pe mulţimea lexicoanelor o structură de graf orientat. Acest lucru se datorează faptului că adesea unele clase morfosintactice (sublexicoane) constituie clase de continuare pentru mai multe clase morfosintactice.

Exemplul 3.3 *Să facem o analiză a cuvântului* **brazi**. *Printre regulile pe care le poate folosi procesul de analiză se află şi regula*

$$R_1: \qquad d{:}z \longleftrightarrow \{V{:}V - < +{:}\emptyset\; i{:}i >\}$$

Prin inserarea unui morfem nul se obţine secvenţa de suprafaţă "braz∅i", care este prelucrată, conducând la secvenţa de perechi simbolice:

$$< b{:}b\; r{:}r\; a{:}a\; d{:}z\; +{:}\emptyset\; i{:}i >$$

În continuare se efectuează separarea folosind ca referinţă delimitatorul morfemic +; se obţin astfel două morfeme "brad" şi "+i", care indică faptul că se analizează un substantiv comun, genul masculin (primul morfem), nearticulat, plural (al doilea morfem).

Evident, ar putea exista şi alte reguli care se pot aplica, dar morfemele rezultate nu ar fi valide (nu e regăsesc în dicţionar). În acest fel se elimină variantele incorecte de translatare spre nivelul lexical.

Invers, la generarea cuvântului "cozi" se pleacă de la rădăcina "coad" (substantiv comun, genul feminin) iar prin adăugarea morfemului "+i" (nearticulat plural) se obţine secvenţa "coad + i". Aplicând acum regula:

$$R_2: \qquad d{:}z \longleftrightarrow \{< o{:}o\; a{:}\emptyset > - < +{:}\emptyset\; i{:}i >\}$$

se obţine următoarea secvenţă de perechi simbolice:

$$< c{:}c\; o{:}o\; a{:}\emptyset\; d{:}z\; +{:}\emptyset\; i{:}i >$$

Aceasta va duce la sintetizarea secvenţei de suprafaţă "co∅z∅i", din care - prin eliminarea morfemelor nule - se obţine forma flexată "cozi".

Se observă că în acest caz se putea aplica şi regula R_1, ceea ce ar fi dus la generarea unui cuvânt incorect; este o exemplificare a uneia din deficienţele modelului.

3.4 Morfologia paradigmatică

Al doilea model morfologic procedural prezentat se înscrie în clasa morfologiilor flexionare şi a fost elaborat de Dr. Dan Tufiş de la Institutul de cercetări al Academiei Române. Marele său avantaj constă în naturaleţea sa - posibilitatea de a fi dezvoltat prin tehnici ale învăţării.

Definiţia 3.12 *Se numeşte model morfologic (MM) structura*

$$(C, SC, A, V, F_1, F_2, F_3, P)$$

unde:

- $C = \{c_1, c_2, \ldots, \bar{c}_i\}$ *este o mulţime de categorii;*
- $SC = \{sc_1, sc_2, \ldots, \jmath\}$ *este o mulţime de subcategorii ale categoriilor din C;*

- $A = \{a_1, a_2, \ldots, a_k\}$ *este o mulţime de atribute ale subcategoriilor din* SC *(în raport cu care fenomenul flexionar este relevant);*

- $V = \{v_1, v_2, \ldots, v_m\}$ *este o mulţime de valori pe care le pot lua atributele din* A*;*

- $F_1 : C \longrightarrow 2^{SC}, F_2 : SC \longrightarrow 2^A, F_3 : A \longrightarrow 2^V$ *sunt aplicaţii;*

- $P \subseteq C \times SC \times 2^A \times 2^V$ *- numit spaţiul paradigmatic flexionar - are proprietatea că* $\forall p_i = (c_i, sc_i, A_i, V_i) \in P$ *sunt adevărate afirmaţiile:*

  1. $c_i \in C, sc_i \in F_1(c_i)$;
  2. $A_i = \{a_{i_1}, a_{i_2}, \ldots, a_{i_k}\} \subset A, V_i = \{v_{i_1}, v_{i_2}, \ldots, v_{i_k}\} \subset V$;
  3. $\exists M_i = \{m_{i_1}, \ldots, m_{i_k}\} \subset F_2(sc_i)$ *cu* $v_{i_q} \in F_3(m_{i_q}), 1 \leq q \leq k$.

Se numeşte *familie tematică (FT)* asociată unui lexem Lx_i mulţimea sortată lexicografic a tuturor cuvintelor obţinute prin flexarea gramaticală a lexemului respectiv: $FT(Lx_i) = \{w_1, w_2, \ldots, w_m\}$.

Să notăm cu $< X >$ un şir arbitrar de litere şi cu $< X >< Y >$ şirul de litere obţinut prin concatenarea şirurilor $< X >$ şi $< Y >$.

Definiţia 3.13 *O mulţime* $FT(Lx_i)$ *este* **regulată** *dacă şi numai dacă există un subşir* $< R_q >$ *de lungime* q *numit rădăcină, comun tuturor cuvintelor din* $FT(Lx_i)$ *astfel încât:*

1. $\forall k, < W > \subset FT(Lx_i) \Longrightarrow < w_k > = < R_q >< t_k >$;

2. $< R_q >$ *este cel mai lung subşir cu proprietatea anterioară;*

3. $q \geq$ *o limită inferioară (constantă întreagă, dependentă de limbă);*

4. $\forall j, 2 \leq j \leq m-1$, *submulţimile* $\{w_1, \ldots, w_j\}, \{w_{j+1}, \ldots, w_m\}$ *au rădăcinile* $< R_{q_1} >$ *respectiv* $< R_{q_2} >$ *unde* $< R_{q_1} >$ *este un subşir (nu neapărat propriu) al lui* $< R_{q_2} >$.

Partea care rămâne dintr-un cuvânt din $FT(Lx_i)$ după îndepărtarea rădăcinii se numeşte *terminaţie* (vom folosi termenul de terminaţie atât pentru desinenţe cât şi pentru sufixe).

Definiţia 3.14 *O familie tematică (FT) a unui lexem* Lx_i *se numeşte* **parţial regulată** *dacă există o partiţie* $FT(Lx_i) = \{FT_1(LX_i), \ldots, FT_k(Lx_i)\}$ *astfel încât:*

- $\bigcup_{j=1}^{k} FT_j(Lx_i) = FT(Lx_i)$;

- $\forall m, n (m \neq n), FT_m(Lx_i) \cap FT_n(Lx_i) = \emptyset$;

- $\forall j, FT_j(Lx_i)$ *este regulată şi* $|FT_j(Lx_i)| > 1$.

Deci, o familie tematică parţial regulată este caracterizată de k rădăcini.

Definiţia 3.15 *Mulţimea terminaţiilor obţinute dintr-o familie tematică (regulată sau parţial regulată) se numeşte familie paradigmatică FP.*

Dacă notăm cu T reuniunea tuturor FP, se obţine noţiunea centrală a modelului, respectiv *paradigma flexionară* Q în modul următor:

- $Q = \{(t_1, p_1), (t_2, p_2), \ldots, (t_k, p_k)\} \subset 2^{T \times P}$;
- $\exists FP_i$ astfel încât $FP_i = \{t_1, t_2, \ldots, t_k\}$.

Pentru a simplifica notaţia, vom folosi în continuare paranteze unghiulare numai când va fi necesar de evidenţiat (des)compunerea unei forme flexate.

Fie FL mulţimea obţinută din reuniunea a k familii tematice, numită *fond lexical*:

$$FL = \bigcup_{j=1}^{k} FT(Lx_i)$$

Vom numi *dicţionar neinterpretat* al fondului lexical FL mulţimea

$$DN = \{R_1, R_2, \ldots, R_p\}$$

cu proprietatea că oricare R_i este o rădăcină a unei familii tematice FT_j din FL.

Aplicaţia $I : FL \longrightarrow 2^{DN \times P}$ se numeşte *interpretarea fondului lexical* FL în modelul morfologic MM în cadrul căruia este definit P. Se observă că interpretarea unui cuvânt poate să nu fie unică, fapt natural la nivelul analizei izolate a cuvântului. Acest lucru se datorează ambiguităţii (intriseci sau extrinseci) legate de omografie. Interpretarea I abstractizează procesul de analiză morfologică.

Abstractizarea procesului invers, de generare a formelor flexate, este reprezentată de aplicaţia $G : DN \times P \longrightarrow FL$.

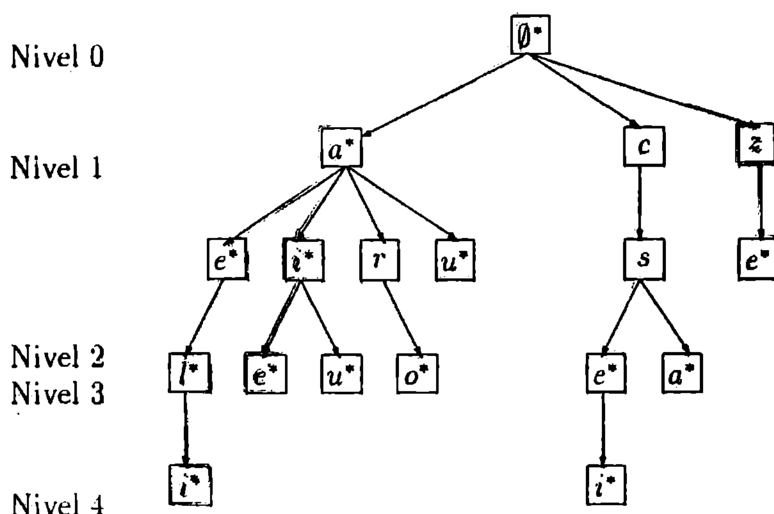
3.4.1 Organizarea terminaţiilor

Conform definiţiilor anterioare, un cuvânt este privit - din punct de vedere structural - ca o juxtapunere a unei părţi fixe (*tema*) şi a unei părţi variabile (*terminaţia*). În cadrul terminaţiei nu se mai face distincţie între sufixe şi desinenţe. Un cuvânt poate coincide cu tema sa, caz în care spunem că are terminaţie nulă. Conceptul de temă are un conţinut lexical, dar unui cuvânt îi pot corespunde mai multe teme, cum sunt de exemplu verbele neregulate.

Terminaţiile sunt organizate arborescent, fiecare nod al arborelui fiind etichetat cu câte o literă care intră în compoziţia unei terminaţii. Un drum care pleacă de la rădăcina arborelui (corespunzătoare terminaţiei nule) la o frunză semnifică una sau mai multe terminaţii obţinute prin concatenarea literelor asociate nodurilor parcurse. Există posibilitatea (în limba română) ca un drum în arbore să reprezinte mai multe terminaţii, care se încheie la diferite noduri ale drumului.

Definiţia 3.16 *Se numeşte* **nod terminal** *orice nod al arborelui de terminaţii cu proprietatea că prin concatenarea literelor asociate nodurilor parcurse pe drumul de la rădăcină la nodul respectiv se obţine o terminaţie validă.*

Tabelul 3.1: Arbore de terminaţii



În Tabelul 3.1 este reprezentat un fragment al arborelui de terminaţii specifice limbii române; nodurile terminale sunt notate cu simbolul *.

Nodurile care apar pe nivelul 1 al arborelui sunt etichetate cu litere ce pot apare pe ultima poziţie a unei terminaţii. Literele care etichetează nivelul 2 în arbore pot apare ca penultime litere în terminaţii, dar numai dacă sunt urmate de literele care etichetează nodurile de pe nivelul 1. Adâncimea arborelui este egală cu lungimea celei mai lungi terminaţii. Deci *ilea, lea, ea, a, eia, uia, ia, ora, ua, iesc, esc, asc, ez* sunt terminaţii valide (citirea se face de la nodurile terminale spre rădăcină).

Nodurile terminale au asociate informaţii referitoare la părţile de vorbire şi mărcile morfologice cărora le sunt caracteristice terminaţiile definite de nodurile respective. Astfel, nodului *l* de pe drumul $\emptyset - a - e - l - i$, care defineşte terminaţia *lea* îi sunt asociate informaţiile *substantiv feminin, articulat enclitic, cazul nominativ/acuzativ* - cum este situaţia în cuvintele *ca<u>lea</u>, pie<u>lea</u>*, şi respectiv *numeral ordinal, masculin* - ca în cuvintele *doi<u>lea</u>, zece<u>lea</u>*.

Terminaţia *ea*, inclusă în *lea* este însă şi mai ambiguă:

- Verb, prezent, conjunctiv, persoana 3, singular/plural;
- Verb, imperfect, persoana 3, singular;
- Verb, infinitiv;
- Substantiv, feminin, articulat enclitic, caz nominativ/acuzativ.

Exemple de cuvinte în care apare terminaţia ce poartă aceste informaţii ar fi: *să st<u>ea</u>, merg<u>ea</u>, a căd<u>ea</u>, cart<u>ea</u>*, etc. De remarcat că pronumele personal *ea* nu poate fi interpretat ca o terminaţie, deoarece tema ar fi nulă lucru neacceptat conform definiţiilor anterioare. Cu cât se urcă în arbore, gradul de nedeterminism creşte; astfel, terminaţiilor *a* şi $\emptyset$ le sunt asociate 33 respectiv 17 interpretări.

3.4.2 Dicţionarul

Din punct de vedere logic, dicţionarul este o colecţie ordonată a mulţimii de reprezentanţi ai fondului lexical prelevat. Un reprezentant al unui element lexical este o colecţie structurată de informaţii, numită *intrare lexicală*. Intrările lexicale sunt deci ordonate logic după unul sau mai multe câmpuri - considerate "*chei*" - ale structurii de reprezentare. Cea mai frecventă cheie este cea corespunzătoare formei cuvântului reţinut în dicţionar, căruia i se ataşează intrarea lexicală: rădăcina, tema sau cuvântul. Fiecare temă prezentă în dicţionar are ataşate una sau mai multe descrieri. O descriere constă dintr-un identificator unic în dicţionar, care corespunde noţiunii de lexem, un cod care identifică partea de vorbire asociată lexemului, o caracterizare morfologică a lexemului, precum şi una sau mai multe descrieri sintactice şi semantico-pragmatice asociate. Formal, o intrare lexicală este de forma dată în Tabelul 3.2.

Tabelul 3.2: Structura unei intrări în dicţionar

< *intrare lexicală* >::= < *temă* >
(< *lexem* >
< *parte de vorbire* >
< *descriere morfologică* >
(< *descriere sintactică* >
(< *descriere semantico-pragmatică* >)*
)*
)$^+$

Unei teme îi pot corespunde mai multe lexeme. De exemplu, pentru tema *munc* vor exista lexemele *a munci* (verb) şi *muncă* (substantiv).

Câmpurile <*parte de vorbire*> şi <*descriere morfologică*> asociate unui lexem sunt unice. Omografele vor fi diferenţiate prin identificatori lexemici diferiţi; de exemplu, pentru tema *duc* care are asociate două lexeme omografe (*duce* - a căra, a transporta, şi *duce* - rang nobiliar), distincţia între lexeme se va face prin intermediul unui indice numeric: *duce1*, *duce2*.

Două sau mai multe teme distincte pot avea asociate acelaşi lexem, situaţie corespunzătoare în dicţionar sinonimiei. De asemenea, polisemia este asigurată prin posibilitatea de a avea mai multe descrieri semantico-pragmatice pentru acelaşi lexem. Informaţiile din dicţionar corespunzţoare acestor câmpuri (sintactic şi semantic) nu sunt folosite în faza analizei morfo - lexicale, ci în fazele ulterioare.

Prelegerea 4

Gramatici şi analizori sintactici

Pentru a formaliza studiul structurii sintactice a unei propoziţii, sunt necesare două noţiuni noi: **gramatica** - o construcţie formală specifică unei structuri lingvistice, şi **tehnicile de analiză sintactică** care determină corectitudinea unei propoziţii în conformitate cu regulile gramaticii.

Vom face observaţia că mare parte din elementele folosite în acest capitol sunt cunoscute din cursurile anilor anteriori ([2], [3]). Vom încerca de aceea să punctăm numai diferenţele de prezentare a conceptelor, cu o apropiere de structura temei actuale - lingvistica matematică.

4.1 Gramatici şi structuri lingvistice

Unul din punctele majore de interes ale lingvisticii matematice constă în descrierea modului de descompunere al unei propoziţii şi explorarea structurilor sale corecte din punct de vedere sintactic. Aparatul folosit în aceste explorări este de obicei *arborele de derivare.*

Un astfel de arbore este reprezentat fie sub forma uzuală (arborele asociat unei derivări independente de context) sau prin operaţia de parantezare.

Exemplul 4.1 *Să considerăm gramatica (independentă de context) cu producţiile:*

1.	$S \longrightarrow NP\ VP$		5.	$NUME \longrightarrow$	*Andrei*
2.	$VP \longrightarrow V\ NP$		6.	$V \longrightarrow$	*a rupt*
3.	$NP \longrightarrow NUME$		7.	$ART \longrightarrow$	*o*
4.	$NP \longrightarrow ART\ N$		8.	$N \longrightarrow$	*muscă*

Producţiile $5-8$ *(de tip* < *neterminal* >⟶< *terminal*>*) se numesc* **producţii lexicale**. *În studiul care urmează, acest tip de reguli vor fi ignorate, deoarece nu modifică cu nimic fondul discuţiei.*

Semnificaţia neterminalelor (definită şi în prelegerile anterioare) este:

S – *propoziţie*	NP – *expresie-substantiv*	VP – *expresie-verb*
ART – *articol*	$NUME$ – *substantiv propriu*	V – *verb*
N – *substantiv comun*		

Pentru propoziţia **Andrei a rupt o muscă** *se poate construi derivarea (stângă):*
$S \Longrightarrow NP\ VP \Longrightarrow NUME\ VP \Longrightarrow$ *Andrei VP* $\Longrightarrow$ *Andrei V NP*
$\Longrightarrow$ *Andrei a rupt NP* $\Longrightarrow$ *Andrei a rupt ART N*
$\Longrightarrow$ *Andrei a rupt o N* $\Longrightarrow$ *Andrei a rupt o muscă.*

Arborele de derivare corespunzător poate fi reprezentat în două moduri:

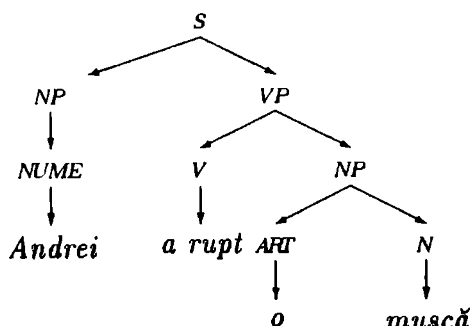


Figura 4.1

sau, sub formă parantezată (parcurgerea infix a unui arbore oarecare):

(*S* (*NP* (*NUME* *Andrei*))
(*VP* (*V* *a rupt*)
(*NP* (*ART o*)
(*N muscă*))))

Să reamintim câteva elemente. Unei gramatici i se pot asocia două modalităţi de operare:

- *Modalitatea generativă*: pentru construirea limbajului asociat se folosesc derivările;
- *Modalitatea de recunoaştere*: se caută structura propoziţiilor care pot fi construite de regulile gramaticii.

Modalităţile de recunoaştere se împart în:

1. Strategii **top-down**: se pleacă de la S şi se caută pe diverse căi să se rescrie simbolurile, până se ajunge la propoziţia căutată (sau până au fost epuizate toate variantele de construcţie - în caz că se caută structura unei propoziţii incorecte din punct de vedere al gramaticii enunţate);
2. Strategii **bottom-up**: se pleacă de la cuvintele propoziţiei şi prin procedee de deplasare-reducere (căutând membrii drepţi ai regulilor şi înlocuindu-i cu membrii stângi corespunzători), se caută construirea arborelui de jos în sus, până se ajunge la simbolul rădăcină S.

4.2 Semnificaţia unei gramatici bune

La construcţia gramaticii care să caracterizeze un limbaj $\mathcal{L}$ peste un alfabet V, trebuiesc urmărite câteva obiective:

- **generalitate**: ordinul de mărime al lui $card(\mathcal{L})$ (propoziţiile pe care gramatica le consideră corecte);
- **selectivitate**: ordinul de mărime al lui $card(V^* - \mathcal{L})$ (propoziţiile pe care gramatica le consideră greşite sintactic);
- **înţelegere**: simplitatea gramaticii înseşi.

În gramaticile cu reguli puţine (cum sunt cele folosite în exemple) sunt generate doar un număr mic de tipuri de propoziţii; aici, o analiză structurală poate părea fără rost; totuşi, analizorii sintactici construiţi pentru aceste gramatici sunt uşor de generalizat la limbaje mari, în timp ce alte module de lucru vor necesita modificări majore.

În generalizarea limbajului prin extensia gramaticii trebuiesc avute în vedere anumite proprietăţi pe care să le aibă noile propoziţii. În particular se va urmări modul în care propoziţia este împărţită în componente numite *constituienţi*. La aceasta, un rol important îl joacă intuiţia şi experienţa, care ajută la construţia diverselor teste specifice.

Să detaliem puţin această afirmaţie.

Ori de câte ori se observă că un grup de cuvinte formează un anumit constituient α, pasul următor este de a construi noi propoziţii care să utilizeze α în legătură cu alte grupe de cuvinte clasificate ca aparţinând aceluiaşi tip de constituienţi. Acest test se bazează pe faptul că în majoritatea cazurilor, numai constituienţii de acelaşi tip pot fi legaţi.

Tabelul 4.1 oferă câteva exemple în acest sens.

Tabelul 4.1:

$NP - NP$	*Eu am mâncat o supă şi o friptură.*
$VP - VP$	*Voi mânca supa şi voi renunţa la friptură.*
$S - S$	*Eu mănânc supa şi Andrei mănâncă friptura.*
$PP - PP$	*Am văzut aceeaşi jucărie în vitrină şi pe raft.*
$ADJP - ADJP$	*Am luat din frigider o friptură rece şi arsă.*
$ADVP - ADVP$	*Rodica mergea pe drum încet şi cu foarte mare atenţie.*
$N - N$	*Eu am mâncat o supă şi o friptură.*
$V - V$	*Maria găseşte şi scoate puloverul din dulap.*
$AUX - AUX$	*Paul poate şi vrea să treacă examenul acesta.*
$ADJ - ADJ$	*Podul trece peste o apă foarte rece şi repede (adică foarte rece şi foarte repede).*

Totuşi, dacă nu suntem atenţi, aceste construcţii pot conduce şi la propoziţii incorecte gramatical, cum ar fi:

Eu am mâncat o friptură încet şi bine prăjită.

Alte teste se construiesc inserând constituienţii propuşi în alte propoziţii care folosesc aceeaşi categorie de constituienţi.

4.3 Algoritm de analiză sintactică top-down

Un algoritm de analiză sintactică poate fi construit foarte simplu ca o procedură care încearcă diverse modalităţi de combinare a formelor gramaticale, în scopul de a ajunge la o combinaţie pe baza căreia se poate construi un arbore de derivare corespunzător structurii secvenţei de intrare.

În prima fază, nu vom fi interesaţi de construcţia arborelui, ci doar de răspunsul (**DA** sau **NU**) dacă este posibilă construcţia unui astfel de arbore (deci dacă secvenţa de intrare poate fi generată sau nu de gramatică).

Algoritmii top-down de analiză sintactică lucrează în manieră predictivă([3]): pleacă de la simbolul de start S şi încearcă să-l rescrie într-o secvenţă de simboluri terminale, consistentă cu secvenţa de intrare.

Starea unui astfel de analizor la un anumit moment poate fi caracterizată de o listă a simbolurilor rezultate în urma operaţiilor de derivare până atunci; această stare o vom numi *listă de simboluri.* De exemplu, analizorul pleacă din starea iniţială S şi, după aplicarea regulii $S \longrightarrow NP\ VP$, lista de simboluri va fi $(NP\ VP)$. Dacă în continuare se aplică producţia $NP \longrightarrow ART\ N$, lista de simboluri va fi $(ART\ N\ VP)$, ş.a.m.d.

Analizorul poate continua în acest mod până ajunge într-o stare formată numai din terminale; în acest moment, el va testa dacă această stare coincide cu secvenţa de intrare.

O asemenea modalitate este însă prea greoaie deoarece, în caz de eroare, necesită întoarceri care sporesc enorm complexitatea algoritmului.

Un algoritm mai bun (deşi nu elimină întoarcerile) este acela în care intrarea este comparată cu secvenţa de terminale obţinută la fiecare pas. În plus, se poate folosi aici o structură numită **lexicon** care stochează eficient pentru fiecare cuvânt, toate categoriile lexicale din care poate face parte.

Exemplul 4.2 *Un lexicon simplu poate fi:*

plânge: V;
câine: N;
Un: ART;

De asemenea, vom folosi gramatica independentă de context:

1. $S \longrightarrow NP\ VP$
2. $NP \longrightarrow ART\ N$
3. $NP \longrightarrow ART\ ADJ\ N$
4. $VP \longrightarrow V$
5. $VP \longrightarrow V\ NP$

Dacă se foloseşte lexiconul de sus, gramatica determină un limbaj în mod neambiguu, chiar dacă nu conţine nici o producţie lexicală.

Cu aceste convenţii, o stare a unui analizor sintactic top-down este definită ca o pereche (α, k) unde α este o listă de simboluri (ca mai sus) iar k este un număr indicând poziţia curentă în secvenţă. Fiecare poziţie reprezintă numărul cuvântului care urmează în secvenţă.

Exemplul 4.3 *Să considerăm propoziţia:*

$_1$ *Un* $_2$ *câine* $_3$ *plânge* $_4$

O stare tipică de analiză poate fi: ((*N VP*) 2).

În ea se indică faptul că analizorul caută un N urmat de un VP, pornind de pe poziţia 2. *Noile stări sunt generate din cele vechi, în funcţie de primul simbol (dacă este simbol lexical sau nu). Dacă acesta este un simbol lexical (cum ar fi N) şi următorul cuvânt din secvenţa de intrare poate aparţine acestei categorii lexicale, starea se poate modifica eliminând primul simbol şi repoziţionând contorul. În propoziţia propusă pentru analiză, deoarece cuvântul "câine" apare în lexicon ca un N, următoarea stare a analizorului va fi* ((*VP*) 3), *ceea ce înseamnă că trebuie găsit un VP plecând de la poziţia* 3.

Dacă primul simbol este un neterminal (cum ar fi VP), acesta este rescris folosind o regulă din gramatică. De exemplu, cu regula 4 *din gramatica dată în exemplul 4.2, noua stare va fi* ((*V*) 3) *iar cu regula* 5 – ((*V NP*) 3).

Un algoritm de analiză sintactică top-down bazat pe acest formalism şi folosind o tehnică backtracking (pentru a nu rata eventuale soluţii) este construit mai jos.

4.3.1 Algoritm de analiză sintactică top-down

În acest caz se construieşte o listă de stări, numită **listă de posibilităţi**. Primul element al listei este starea curentă, (sub forma definită anterior); celelalte elemente sunt *stări backup* (de revenire), fiecare indicând o alternativă apărută pe parcursul derivării.

Exemplul 4.4 *Lista de posibilităţi:*

(((*N*) 2)((*NUME*) 1)((*ADJ N*) 1))

indică faptul că avem starea curentă formată din lista de simboluri (*N*) *din poziţia* 2, *şi două backup-uri posibile, unul format din lista* (*NUME*) *din poziţia* 1, *celălalt din lista de simboluri* (*ADJ N*), *de asemenea din poziţia* 1.

Algoritmul pleacă din starea iniţială ((S) 1), fără stări backup. Paşii lui sunt:

1. Selectează starea curentă: Se scoate prima stare din lista de posibilităţi; să o numim C. Dacă lista de posibilităţi este vidă, secvenţa de intrare este respinsă (nu este conformă cu gramatica limbajului) şi **STOP**.

2. Dacă C are o listă de simboluri vidă şi poziţia cuvântului este la sfârşitul secvenţei, aceasta este acceptată ca fiind corectă; **STOP**.

3. În celelalte cazuri se generează următoarele stări posibile (care se pun la începutul listei de posibilităţi):

   (a) Dacă primul element al listei de simboluri din C este un simbol lexical, iar următorul cuvânt din secvenţă poate fi în clasa acestui simbol, se crează o nouă stare eliminând primul simbol din lista de simboluri şi repoziţionând contorul.

(b) Dacă primul simbol din lista de simboluri din C este un neterminal, se generează o nouă stare pentru fiecare regulă din gramatică care poate rescrie acest neterminal.

Exemplul 4.5 *Să aplicăm acest algoritm pentru gramatica de mai sus şi pentru secvenţa de intrare*

Un câine plânge.

Vom avea următorii paşi:

Pas	*Stare curentă*	*Stări backup*	*Comentarii*
1.	((*S*) 1)		*Poziţia iniţială*
2.	((*NP VP*) 1)		*Se rescrie S cu regula* 1
3.	((*ART N VP*) 1)	((*ART ADJ N VP*) 1)	*Se rescrie NP cu regulile* 2 *şi* 3
4.	((*N VP*) 2)	((*ART ADJ N VP*) 1)	*Se reduce ART cu "Un"*
5.	((*VP*) 3)	((*ART ADJ N VP*) 1)	*Se reduce N cu "câine"*
6.	((*V*) 3)	((*V NP*) 3) ((*ART ADJ N VP*) 1)	*Se rescrie VP cu regulile* 4 *şi* 5
7.	(() 4)	((*V NP*) 3) ((*ART ADJ N VP*) 1)	*Se reduce V cu "plânge"*
8.	**STOP**		*Propoziţie corectă.*

Exemplul 4.6 *Să considerăm acelaşi algoritm şi secvenţa de intrare*

$_1$ *Un* $_2$ *bătrân* $_3$ *toc* $_4$ *scârţâie* $_5$.

Pentru ea, lexiconul va fi:

un : *ART*
toc : *N*, *V*
batrân : *ADJ*, *N*
scârţâie : *V*

Apar aici ambiguităţi din cauza cuvintelor "toc" - care poate fi substantiv dar şi verb, în propoziţii ca

Eu toc frunză la câini.

şi "bătrân" - care poate fi atât adjectiv cât şi substantiv).
Algoritmul de analiză va parcurge paşii din Tabelul 4.2:

Este util de formalizat acest algoritm de analiză sintactică top-down (în maniera algoritmilor predictivi).

Tabelul 4.2:

Pas	*Stare curentă*	*Stări backup*	*Comentarii*
1.	((*S*) 1)		*Poziţia iniţială*
2.	((*NP VP*) 1)		*Se rescrie S cu regula* 1
3.	((*ART N VP*) 1)	((*ART ADJ N VP*) 1)	*Se rescrie NP cu regulile* 2 *şi* 3
4.	((*N VP*) 2)	((*ART ADJ N VP*) 1)	*Se reduce ART cu "Un"*
5.	((*VP*) 3)	((*ART ADJ N VP*) 1)	*Se reduce N cu "bătrân"*
6.	((*V*) 3)	((*V NP*) 3) ((*ART ADJ N VP*) 1)	*Se rescrie VP cu regulile* 4 *şi* 5
7.	((),4)	((*V NP*) 3) ((*ART ADJ N VP*) 1)	*Se reduce V cu "toc"*
8.	((*V NP*) 3)	((*ART ADJ N VP*) 1)	*Se alege primul backup*
9.	((*NP*) 4)	((*ART ADJ N VP*) 1)	
10.	((*ART N*) 4)	((*ART ADJ N*) 4) ((*ART ADJ N VP*) 1)	*Eşec; la* 4 *nu este ART*
11.	(*ART ADJ N*) 4)		*Din nou eşec*
12	((*ART ADJ N VP*) 1)		*Se ia ultimul backup*
13.	((*ADJ N VP*) 2)		*Se reduce ART cu "Un"*
14.	((*N VP*) 3)		*Se reduce ADJ cu "bătrân"*
15.	((*VP*) 4)		*Se reduce N cu "toc"*
16.	((*V*) 4)	((*V NP*) 4)	
17.	(() 5)		*Secvenţă corectă !*

4.4 Algoritm de analiză sintactică bottom-up

După cum se ştie ([3]), algoritmii bottom-up diferă de cei top-down prin modul de folosire a producţiilor. Operaţia de bază este aceea de a lua o secvenţă de simboluri şi a o compara cu membrii drepţi ai producţiilor; dacă există o regulă cu acest membru drept, secvenţa se înlocuieşte cu membrul stâng al producţiei folosite, după care procesul se reia.

Într-un astfel de analizor, o stare este o listă de simboluri, iniţial fiind secvenţa de analizat. Stările succesoare pot fi generate explorând toate posibilităţile astfel ca:

1. Să se rescrie un cuvânt din secvenţă prin toate categoriile sale lexicale posibile;

2. Să se înlocuiască o secvenţă de simboluri care formează membrul drept al unei reguli, cu membrul ei stâng.

Din păcate, o astfel de implementare conduce la un algoritm foarte ineficient ca timp; pentru a simplifica problema, vom introduce o structură de date numită **hartă**, în care se stochează rezultatele parţiale ale procesului de identificare (pentru a elimina repetarea lor).

Procesul de identificare are loc pe baza unui constituient numit **cheie**; de fiecare dată se caută identificarea pentru producţiile al căror membru drept începe cu cheia, sau au început cu o cheie anterioară şi solicită o nouă cheie pentru a completa membrul drept al producţiei sau a o extinde la una nouă.

Înainte de a formaliza procedeul, să luăm un exemplu.

Exemplul 4.7 *Fie gramatica de producţii:*

1. *S* ⟶ *NP VP*	3. *NP* ⟶ *ART N*	5. *VP* ⟶ *AUX VP*
2. *NP* ⟶ *ART ADJ N*	4. *NP* ⟶ *ADJ N*	6. *VP* ⟶ *V NP*

Să presupunem că analizăm o secvenţă care începe cu un ART. Cu acest ART drept cheie pot fi activate producţiile 2 *şi* 3 *(sunt singurele care încep cu un ART). Pentru a semnala aceasta la analiza următoarei chei, trebuie observat că regulile* 2 *şi* 3 *pot fi continuate din punctul de după ART; vom marca acest fapt scriind producţia cu un semn (•) pentru a marca de unde se continuă analiza. Deci, vom înregistra:*

2′. *NP* ⟶ *ART* • *ADJ N*

3′. *NP* ⟶ *ART* • *N*

Dacă următoarea cheie de intrare este un ADJ, atunci poate începe regula 4 *(ca mai sus), iar regula* 2′ *poate fi extinsă până la:*

2". *NP* ⟶ *ART ADJ* • *N*

Harta menţine înregistraţi toţi constituienţii derivaţi din secvenţă, precum şi toate producţiile folosite parţial în derivare; acestea sunt numite **arcuri active**. *De exemplu, după ce s-a depistat un ART iniţial urmat de un ADJ, se ajunge la harta următoare:*

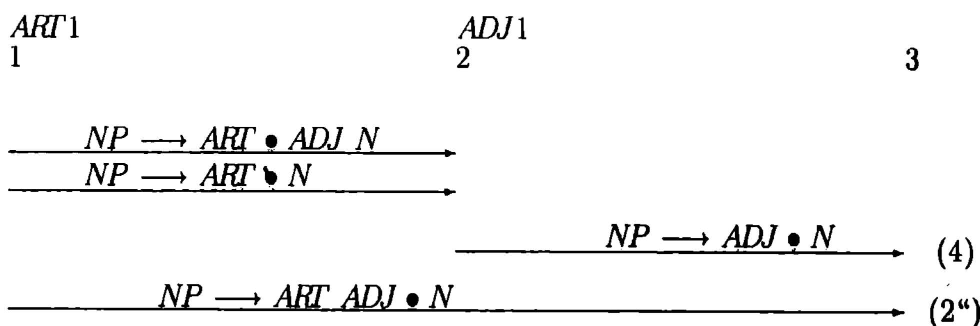


Figura 4.2

Interpretarea Figurii 4.2 *este următoarea: există pe hartă doi constituienţi compleţi: ART*1 *de la poziţia* 1 *la poziţia* 2, *şi ADJ*1 *de la poziţia* 2 *la poziţia* 3. *Sunt* 4

arce active indicând constituienții posibili; ele sunt indicate prin săgeți și interpretate astfel (de sus în jos):

- *Există un NP posibil cu plecare din poziția 1, care necesită un ADJ începând cu poziția 2.*
- *Este posibil un NP care pleacă din poziția 1, și cere un N pe poziția 2.*
- *Este posibil un NP care pleacă din poziția 2 și cere un N pe poziția 3.*
- *În sfârșit, mai este posibil un NP care pleacă din poziția 1 cu un ART urmat de un ADJ și cere un N începând cu poziția 3.*

Operațiile de bază ale unui analizor bazat pe hărți folosesc o combinare între arce active și constituienți completi. Rezultatul este sau un constituient complet nou sau un arc activ nou, extensie a arcului activ originar. Noii constituienți activi completi sunt depuși într-o listă numită **agendă**, până în momentul când sunt adăugați la hartă.

Procesul se poate exprima algoritmic astfel:

Algoritmul de extensie a arcelor:

Pentru a adăuga un constituient C de la poziția p_1 la poziția p_2:

1. Se inserează C în hartă de la p_1 la p_2;
2. Pentru orice arc activ de forma $X \longrightarrow X_1 \ldots \bullet C \ldots X_n$ de la p_0 la p_1, se adaugă un arc activ nou $X \longrightarrow X_1 \ldots C \bullet \ldots X_n$ de la p_0 la p_2;
3. Pentru orice arc activ de forma $X \longrightarrow X_1 \ldots X_n \bullet C$ de la p_0 la p_1, se adaugă în agendă un nou constituient de tipul X de la p_0 la p_2.

Fiind dat acest algoritm, putem construi un algoritm de analiză bottom-up astfel:

Se efectuează următorii pași până se parcurge toată secvența de intrare:

1. Dacă agenda este vidă, se caută și se introduc în agendă toate interpretările pentru următorul cuvânt de la intrare;
2. Se selectează un constituient C din agendă, de la poziția p_1 la poziția p_2;
3. Pentru fiecare regulă din gramatică de forma $X \longrightarrow C X_1 \ldots X_n$ se trasează un arc activ de forma $X \longrightarrow C \bullet X_1 \ldots X_n$ de la p_1 la p_2;
4. Se adaugă C la hartă folosind algoritmul de extensie a arcelor.

Exemplul 4.8 : *Să luăm următoarea secvenţă de intrare*

O vie vie are rod.

Vom folosi pentru analizor gramatica de producţii:

1. $S \longrightarrow NP\ VP$ 3. $NP \longrightarrow ART\ ADJ\ N$ 5. $VP \longrightarrow V\ N$
2. $NP \longrightarrow ART\ N$ 4. $VP \longrightarrow V$

plus lexiconul:

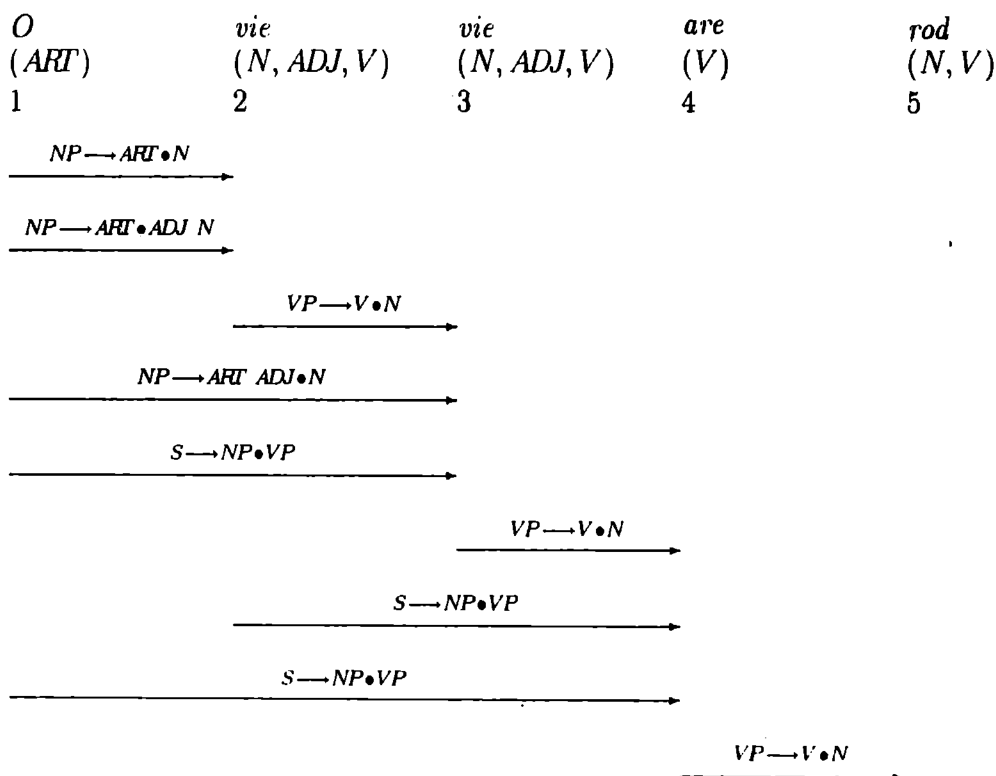
O:ART
vie:N, ADJ, V
are:V
rod:N, V

*Iniţial agenda este vidă; deci se citeşte cuvântul "O" şi se plasează în agendă constituientul ART*1. *Deci:*

- *Se introduce ART*1 *("O" de la* 1 *la* 2*);*
- *Se adaugă arcul activ* $NP \longrightarrow ART \bullet ADJ\ N$ *de la* 1 *la* 2*;*
- *Se adaugă arcul activ* $NP \longrightarrow ART \bullet N$ *de la* 1 *la* 2*;*

(ambele activităţi au fost generate de pasul 3 *al algoritmului, pe baza producţiilor* 2 *şi* 3 *ale gramaticii).*

În continuare analizorul se desfăşoară după harta următoare:



Prelegerea 5

Alte modele gramaticale

5.1 Reţele de tranziţie

În afara formalismului de reprezentare a structurii sintactice - folosind gramatici independente de context, se mai poate utiliza (şi aplica cu succes în analiza sintactică) o altă reprezentare grafică. Ea se numeşte *reţea de tranziţie* şi este o metodă de recunoaştere (spre deosebire de gramatici, care sunt metode generative).

Definiţia 5.1 *O reţea de tranziţie este un graf orientat, cu arcele marcate, în care există un nod (stare) iniţial şi un set de noduri finale. Marcajul nodului iniţial dă numele reţelei. Arcele sunt marcate cu categorii sintactice.*

Exemplul 5.1 *Fie reţeaua numită NP reprezentată prin graful din Figura 5.1*
Ea caracterizează acelaşi limbaj ca şi gramatica de producţii:

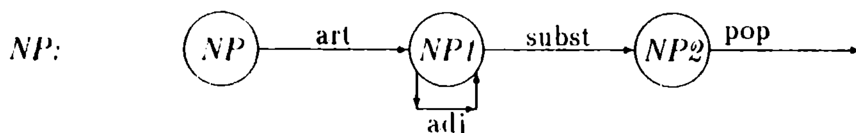
$$NP \Longrightarrow ART\ NP1, \quad NP1 \Longrightarrow ADJ\ NP1, \quad NP1 \Longrightarrow N$$

Modul de utilizare este următorul. Se pleacă din starea iniţială; fiind într-un nod P, se traversează un arc care pleacă din P numai dacă acel arc este marcat cu o categorie sintactică care reprezintă cuvântul curent din secvenţa de intrare. După parcurgerea unui arc, următorul cuvânt din secvenţă devine cuvânt curent.

O frază este *corectă sintactic* dacă există un drum de la nodul iniţial la un nod final (din care iese un arc marcat cu *pop*) care poate fi parcurs de frază.

Evident, reţelele de tranziţie sunt echivalente cu AFD-urile (automatele finite deterministe). Din acest motiv, ele nu pot acoperi toată clasa limbajelor independente de context. Pentru a realiza aceasta, se extinde definiţia la *reţele recursive de tranziţie* (RTN).

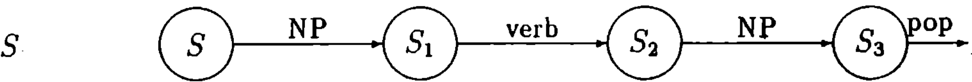
Figura 5.1:



Definiţia 5.2 *O reţea recursivă de tranziţie este o reţea de tranziţie în care mulţimea marcajelor arcelor se îmbogăţeşte şi cu nume de reţele de tranziţie.*

Exemplul 5.2 *O reţea RTN care acceptă propoziţii simple din limba română este:*

Figura 5.2:



Marcajele scrise cu litere mari se referă la reţele. Arcul de la S la S_1 poate fi traversat numai dacă reţeaua NP din Figura 5.1 poate fi traversată până la un arc marcat 'pop'.

Evident, o reţea *RTN* poate conţine arce marcate cu propriul ei nume (deci o definiţie recursivă). În practică, sistemele *RTN* încorporează unele tipuri adiţionale de arce, utile dar nu formal necesare. Aceste tipuri sunt (Tabelul 5.1):

Tabelul 5.1:

Tip de arc	Semnificaţie
CAT	Este traversat numai de un cuvânt din categoria specificată de marcaj
WRD	Este traversat numai de un cuvânt identic cu marcajul
PUSH	Este un arc marcat cu numele unei reţele
JUMP	Este un arc care poate fi traversat fără nici o condiţie
POP	Arc care semnalează sfârşitul reţelei

5.2 Analiză sintactică top-down folosind *RTN*

Algoritmul prezentat este similar celui pentru gramatici independente de context ([2]). Astfel, starea analizorului poate fi reprezentată sub forma unui triplet

$$(n_c, p_c, rev, s_rev)$$

unde:

- n_c (**nod curent**) este nodul din reţea, activ în momentul acţiunii;
- p_c (**poziţie curentă**) este un pointer la următorul cuvânt de analizat din secvenţa de intrare;
- rev (**puncte de revenire**) este o stivă cu nodurile din alte reţele, unde se revine dacă în reţeaua curentă se parcurge un arc *pop*.

- s_rev (**stăre de revenire**) conţine o altă stare, construită atunci când analizorul a avut la dispoziţie mai multe variante de continuare.

Configuraţia iniţială este $(S, 1, \Lambda, \Lambda)$.
Să presupunem că la un moment dat analizorul se află în starea curentă

$$(n_c, p_c, rev, s_rev)$$

. Atunci el va părăsi nodul curent n_c şi va traversa un arc în următoarele situaţii:

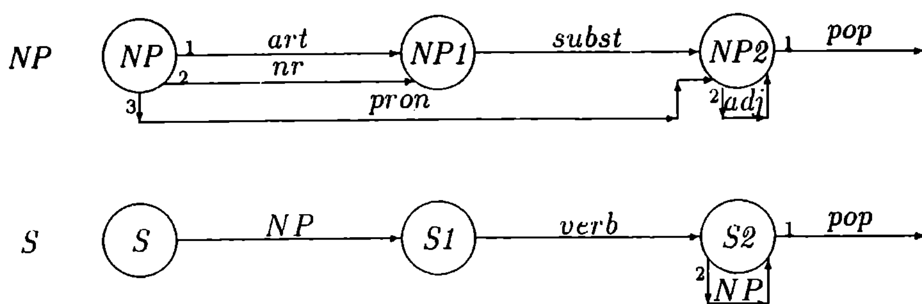
1. Dacă următorul cuvânt din secvenţa de intrare (marcat de p_c) aparţine categoriei sintactice care marchează arcul, atunci
   (a) poziţionează p_c la următorul cuvânt de intrare;
   (b) scrie nodul destinaţie al arcului în locul lui n_c;
   (c) dacă s-a analizat un cuvânt cu mai multe categorii sintactice şi mai există alte arce posibile (corespunzătoare acelor categorii) care pleacă din n_c, se ignoră în reţea arcul prelucrat şi se generează similar pe baza aceluiaşi algoritm o nouă stare, care se introduce în s_rev;
2. Dacă arcul este marcat cu numele unei reţele R atunci
   (a) se introduce nodul destinaţie al arcului în rev;
   (b) se ia nodul iniţial al reţelei R drept nod curent;
3. Dacă arcul este *pop* şi stiva rev nu este vidă, atunci se scoate nodul din topul stivei şi se trece drept n_c;
4. Dacă arcul este *pop*, rev este vidă şi nu mai sunt cuvinte în secvenţa de intrare, atunci *"Analiza este corectă."*
5. Dacă din n_c nu există nici un arc corespunzător categoriei sintactice a cuvântului aflat la p_c, se cercetează s_rev şi:
   (a) dacă $s_rev = \Lambda$, *"Propoziţia nu este corectă."*, Stop.
   (b) dacă $s_rev \neq \Lambda$, se ia drept stare curentă starea din s_rev.

Exemplul 5.3 *Să considerăm limbajul caracterizat de reţeaua din Figura 5.3:*
Numerele scrise pe arce indică ordinea de prioritate a arcelor atunci când sunt mai multe arce care pleacă dintr-un nod.
Folosind aceste reţele, să facem analiza sintactică a propoziţiei

$$_1\ Un\ _2\ câine\ _3\ slab\ _4\ latră\ _5\ doi\ _6\ copii\ _7$$

Lexiconul folosit este următorul:
un: *ART(art), NUM(nr);*
doi: *NUM(nr);*
câine: *N(subst);*
copil: *N(subst);*

Figura 5.3:



a lătra: *V(verb);*
slab: *ADJ(adj);*
(am scris în paranteză notaţiile folosite pentru arcele din reţea).
$(S,1,\Lambda,\Lambda) \vdash (NP,1,S1,\Lambda) \vdash (NP1,2,S1,\Lambda) \vdash (NP2,3,S1,\Lambda) \vdash$
$\vdash (S1,3,\Lambda,(NP1,3,S1,\Lambda)) \vdash (NP2,3,S1,\Lambda) \vdash (NP2,4,S1,\Lambda) \vdash$
$\vdash (S1,4,\Lambda,(NP2,4,S1,\Lambda)) \vdash (S2,5,\Lambda,(NP2,4,S1,\Lambda)) \vdash$
$\vdash (NP,5,S2,(NP2,4,S1,\Lambda)) \vdash (NP1,6,S2,(NP2,4,S1,\Lambda)) \vdash$
$\vdash (NP2,7,S2,(NP2,4,S1,\Lambda)) \vdash (S2,7,\Lambda,(NP2,4,S1,\Lambda)) \vdash$
$\vdash$ *Analiza este corectă.*

5.3 Utilizarea automatelor finite în prelucrarea morfologică

Să reluăm ideea pe baza căreia s-a definit morfologia paradigmatică.

Deşi în exemplele simple şi sistemele mici listarea tuturor cuvintelor din dicţionar este relativ simplă, pentru sistemele cu vocabulare mari, reprezentarea lexiconului reprezintă o problemă dificilă. Nu numai că există un mare număr de cuvinte, dar - după cum am văzut într-o prelegere anterioară - fiecare din ele, combinat cu afixe, produce o serie de alte cuvinte care au semnificaţii diferite. O modalitate des uzitată de a rezolva problema constă în rearanjarea secvenţei de intrare într-o secvenţă de morfeme. Reamintim că un cuvânt poate fi format dintr-un singur morfem, dar poate consta dintr-o rădăcină la care se adaugă diverse afixe (prefixe şi/sau sufixe). De exemplu, la rădăcina *cred* se pot adăuga diverse afixe din mulţimea {*e,em,in,ţă, re, cios*} formându-se cuvinte cum ar fi

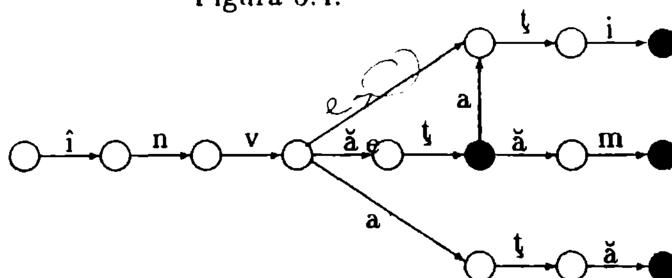
cred-e, cred-in-ţă, cred-in-cios, în-cred-e-re, ş.a.m.d.

O modalitate de reţinere eficientă a cuvintelor cu aceeaşi rădăcină este construirea lor pe baza unor automate finite.

Să luăm de exemplu verbul a *învăţa*. Formele la care se prezintă el la indicativ prezent {*învăţ, înveţi, învaţă, învăţăm, învăţaţi*} reprezintă limbajul acceptat de automatul finit din în Figura 5.4.

El pleacă de la morfemul *vaţ*, în care litera din mijloc se poate modifica în *ă* sau *e*, în funcţie de declinare. După cum se vede, au fost reţinute doar 14 caractere

Figura 5.4:



în loc de 32; informaţiile suplimentare legate de structura automatului (arce, stări finale, etc) sunt uşor de reţinut prin forma de memorare a datelor (de exemplu, liste înlănţuite).

Automatul poate fi completat pentru a lega şi celelalte forme flexionare ale verbului *a învăţa*, precum şi cuvintele înrudite, cum ar fi substantivele care pleacă de la *învăţătură*, adjective (*învăţat*), dar şi forme cu alte prefixe, cum este de exemplu *dezvăţ*.

Întregul lexicon poate fi codificat astfel ca un automat finit (eventual cu ϵ-mişcări) care cuprinde toate cuvintele corecte, şi le transformă în secvenţe de morfeme. Pot fi combinate familii de cuvinte cu afixe comune (cum ar fi de exemplu *învechit, învăţat, înapoiat, înrudit*, etc), dar şi cu porţiuni comune (*untdelemn, candelă, sfredel..*).

Dicţionarele limbilor naturale pot fi uşor modelate în acest fel. În funcţie de lexicon şi de abilitatea de combinare, se poate construi un automat finit cu număr minim de stări.

5.4 Gramatici şi programare logică

O modalitate clară de implementare a algoritmilor de analiză sintactică este folosirea limbajului *PROLOG*.

Exemplul 5.4 *Fie gramatica de producţii:*

1. $S \Longrightarrow NP\,VP$	3. $NP \Longrightarrow N$	5. $VP \Longrightarrow V$	7. $VP \Longrightarrow V\ PP$
2. $NP \Longrightarrow ART\ N$	4. $PP \Longrightarrow P\ NP$	6. $VP \Longrightarrow V\ NP$	

Reprezentarea sa în PROLOG este dată de Tabelul 5.2:
Să presupunem că secvenţa de intrare este:

Andrei cumpără o carte.

Această intrare este descrisă prin:

```
word(Andrei,1,2)
word(cumpara,2,3)
word(o,3,4)
word(carte,4,5)
```

Lexiconul este definit prin mulţimea de predicate, astfel:
isart(o)

Tabelul 5.2:

1.	$s(P1, P3)$	:	$- np(P1, P2), vp(P2, P3)$
2.	$np(P1, P3)$	:	$- art(P1, P2), n(P2, P3)$
3.	$np(P1, P2)$	:	$- nume(P1, P2)$
4.	$pp(P1, P3)$	:	$- p(P1, P2), np(P2, P3)$
5.	$vp(P1, P2)$	:	$- v(P1, P2)$
6.	$vp(P1, P3)$	:	$- v(P1, P2), np(P2, P3)$
7.	$vp(P1, P3)$	:	$- v(P1, P2), pp(P2, P3)$

Tabelul 5.3:

Pas	**Stare curentă**	**Stări backup**	**Comentarii**
1	$s(1,5)$		
2	$np(1, P2)\ vp(P2, 5)$		
3	$art(1, P2')\ n(P2', P2)$ $vp(P2, 5)$	$nume(1, P2)\ vp(P2, 5)$	*Eşec; nu există* **art** *pe poziţia*1
4	$nume(1, P2)\ vp(P2, 5)$		
5	$vp(2, 5)$		*Se verifică* $nume(1, 2)$
6	$v(2, 5)$	$v(2, P2)\ np(P2, 5)$ $v(2, P2)\ pp(P2, 5)$	*Eşec; nu există* **verb** *pe poziţiile* $2-5$.
7	$v(2, P2)\ np(P2, 5)$	$v(2, P2)\ pp(P2, 5)$	
8	$np(3, 5)$	$v(2, P2)\ pp(P2, 5)$	*Se verifică* $v(2, 3)$
9	$art(3, P2)\ n(P2, 5)$	$nume(3, 5)$ $v(2, P2)\ pp(P2, 5)$	
10	$n(4, 5)$	$nume(3, 5)$ $v(2, P2)\ pp(P2, 5)$	*Se verifică* $art(3, 4)$
11	*Succes !*	$nume(3, 5)$ $v(2, P2)\ pp(P2, 5)$	*Se verifică* $n(4, 5)$

isnume(Andrei)
isverb(cumpara)
issubst(carte)

Cuvintele ambigui (pentru care sunt definite mai multe categorii sintactice) vor necesita aserţiuni multiple - câte una pentru fiecare categorie sintactică căreia îi aparţine.

Pentru fiecare categorie sintactică se defineşte câte un predicat, care ia valoarea "Adevăr" dacă şi numai dacă cuvântul dintre poziţiile specificate este din categoria respectivă. Deci:

```
n(I,O)     : - word(Word,I,O),issubst(Word)
art(I,O)   : - word(Word,I,O),isart(Word)
v(I,O)     : - word(Word,I,O),isverb(Word)
nume(I,O)  : - word(Word,I,O),isnume(Word)
```

Algoritmul de analiză sintactică bottom-up este activat acum prin execuţia comenzii $s(1,5)$. *Scriind într-o formă mai detaliată procedura de analiză, se parcurg paşii listaţi în Tabelul 5.3.*

Se pot efectua diverse simplificări ale programelor *PROLOG*, care să mărească eficienţa analizorului (de remarcat că în această scriere, complexitatea algoritmului este exponenţială: C^n - unde C este numărul de reguli iar n - lungimea secvenţei de intrare).

5.5 Definirea formală a unui "Speller Checker"

Elementele fundamentale ale acestei secţiuni sunt prezentate pe larg în [5] şi constituie baza de construcţie a unui controller gramatical pentru limba cehă; ele pot fi transferate fără modificări şi în cazul limbii române.

Definiţia 5.3 *O gramatică independentă de context ne-proiectivă(NCFDG) este o structură* (V_N, V_T, S, P) *cu semnificaţiile obişnuite, unde* P *conţine reguli de rescriere de forma* $A \rightarrow_L BC$, $A \rightarrow_R BC$, *cu* $A \in V_N, B, C \in V$ $(V = V_N \cup V_T)$.

Relaţia de derivare imediată este definită prin:

$$\alpha A\beta\gamma \Longrightarrow \alpha B\beta C\gamma, \quad \text{dacă} \quad A \rightarrow_L BC \in P;$$

$$\alpha\beta A\gamma \Longrightarrow \alpha B\beta C\gamma, \quad \text{dacă} \quad A \rightarrow_R BC \in P.$$

Relaţia de derivare $\Longrightarrow^*$ este închiderea reflexivă şi tranzitivă a relaţiei $\Longrightarrow$. Limbajul $L(G)$ generat de o gramatică G este definit în mod obişnuit.

Observaţie: Pentru $\beta = \epsilon$ se obţine derivarea obişnuită din gramaticile independente de context.

Definiţia 5.4 *Pentru un cuvânt* $w = a_1a_2\dots a_n \in V_T^*$, *se defineşte un arbore* Tr *dominat de nodul* $X \in V$ *astfel:*

1. *fiecare nod din* Tr *este un triplet* $[A,i,j]$ *unde* $A \in V, 1 \leq i,j \leq n$. *Numărul* i *este indexul orizontal al nodului, iar* j *este indexul vertical.*
2. *Nodul* $[A,i,m]$ *are descendenţii* $[B,j,p],[C,k,r]$, *dacă şi numai dacă:*
   - $j < k, m = p + 1, j = i,$ *şi* $A \rightarrow_L BC \in P$, *sau*
   - $j < k, m = r + 1, k = i,$ *şi* $A \rightarrow_R BC \in P$.
3. *Rădăcina este un nod de forma* $[X,i,m]$ *pentru anumiţi* $i,m \in \{1,2,\dots,n\}$.
4. *Nodurile terminale sunt toate nodurile de forma* $[a_i,i,1], 1 \leq i \leq n$.

Pentru simplificare, dacă arborele este dominat de S, nu vom mai specifica aceasta în mod explicit.

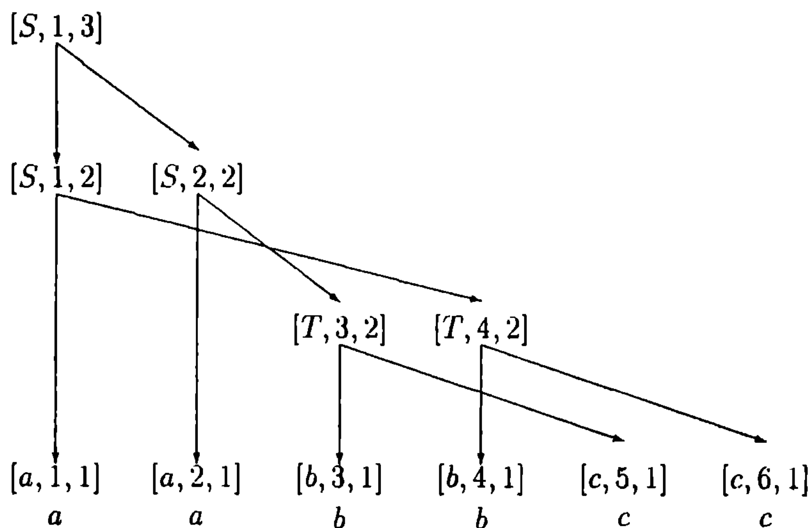
Exemplul 5.5 *Limbajul* $L = \{w|w \in \{a,b,c\}^*, N_a(w) = N_b(w) = N_c(w)\}$ *este generat de o gramatică NCFDG în care* $V_N = \{S,T\}$, $V_T = \{a,b,c\}$, $P = \{S \rightarrow_L aT|Ta|SS,\ T \rightarrow_L bc|cb\}$.

De exemplu, $w = aabbcc$ *este generat prin*

$$S \Longrightarrow_L SS \Longrightarrow_L aST \Longrightarrow_L aaTT \Longrightarrow_L aabTc \Longrightarrow_L aabbcc$$

care are arborele de derivare Tr reprezentat în Figura 5.5:

Figura 5.5:

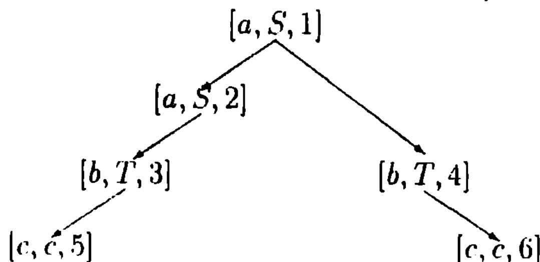


Definiţia 5.5 *Fie Tr arborele corespunzător cuvântului* $w = a_1a_2\ldots a_n \in V_T^*$ *în gramatica NCFDG G. Arborele de dependenţă* $Dep(Tr) = (\Gamma, U)$ *asociat lui Tr se defineşte astfel:*

$\Gamma = \{[a,i] \mid [a,i,1] \in Tr\}$,

U este format din toate arcele $([a,i],[b,j])$ *cu proprietatea că* $[a,i,1]$, $[b,j,1]$ *sunt noduri distincte în Tr, unde există şi un arc* $([A,i,p], [B,j,r])$, *pentru anumiţi* A, B, p, r.

Astfel, pentru arborele din Exemplul 5.5, arborele de dependenţă este (pentru uşurin-ţă, s-a marcat şi neterminalul corespunzător nodului):



Pentru aplicaţia la care ne referim în acest paragraf, trebuiesc construiţi - pentru un cuvânt dat $w = a_1a_2\dots a_n$ - toţi arborii corespunzători dintr-o gramatică *NCFDG G* dată. Algoritmii lucrează cu liste D de itemi de forma

$$D_i = [\ simbol,\ poziţie,\ acoperire,\ ls,\ rs,\ regulă]$$

fiecare item corepunzând derivării unei anumite subsecvenţe a lui w în gramatica G. Semnificaţiile elementelor lui D_i sunt:

- ***simbol*** reprezintă simbolul rădăcină al arborelui de derivare;
- ***poziţie*** este indexul orizontal;
- *acoperire* este mulţimea indicilor orizontali ai nodurilor sale terminale;
- *ls, rs* sunt indicii itemilor D_{ls}, D_{rs} conţinând descendenţii stâng respectiv drept ai arborelui Tr;
- *regulă* este numărul regulii care a generat itemul.

Algoritmul va adăuga la listă cât mai mulţi itemi posibili.

Iniţial lista D va conţine itemii

$$[a_i, i, \{i\}, 0, 0, 0], 1 \le i \le n.$$

Fie $|D|$ numărul de elemente din D.

Derivarea este realizată comparând treptat toate perechile $(D_i, D_j), i, j \in \{1, 2, \dots, |D|\}$ şi cercetând dacă există vre-o regulă $A \to_X D_i.simbol D_j.simbol, (X \in \{R, L\})$; dacă acest lucru este posibil, se crează un nou item care moşteneşte poziţia din D_i sau D_j (după cum regula a fost de tip L respectiv R). Noua acoperire este reuniunea ambelor acoperiri anterioare.

Doi itemi pot genera alt item numai dacă acoperirile lor sunt disjuncte.

```
Algoritmul 1 (Analiză):
for i := 1 to n do
     D_i := [a_i, i, {i}, 0, 0, 0];
Numar_itemi := n;
i := 2
while i ≤ Numar_itemi do
     for j := 1 to i − 1 do
          if D_i.acoperire ∩ D_j.acoperire = ∅ then
               for regula := 1 to Numar_reguli do
               begin
                    Verifica(D_i, D_j, regula)
                    Verifica(D_j, D_i, regula)
               end;
Verifica(D_x, D_y, r)
if r : A →_L D_x.simbol D_y.simbol then
     D := D ∪ {[A, D_x.poziţie, D_x.acoperire ∪ D_y.acoperire, x, y, r]}
else if r : A →_R D_x.simbol D_y.simbol then
     D := D ∪ {[A, D_y.poziţie, D_x.acoperire ∪ D_y.acoperire, x, y, r]}
```

Algoritmul 2 (Recunoaştere):
Este identic cu **Algoritmul 1**, cu o condiţie suplimentară în subalgoritmul *Verifica*: un item nou $[a, b, c, i, j, k]$ se introduce în D numai dacă

$$\forall m, p, q, r \in [1, n],\ [a, m, c, p, q, r] \notin D.$$

Pentru controlul corectitudinii gramaticale a limbajului se vor folosi de fapt două gramatici: una independentă de context, ale cărei reguli ("pozitive") vor genera propoziţii corecte din limbaj. A doua gramatică - *NCFDG* - va conţine reguli (numite "negative") care descriu construcţiile gramaticale incorecte cunoscute, precum şi alte reguli generale referitoare la erori de lexic sau de sintaxă. Această gramatică include ca submulţime strictă mulţimea producţiilor primei gramatici.

Aceste două seturi de reguli pot fi aplicate de algoritmul de analiză anterior, obţinându-se trei rezultate posibile:

- Propoziţia este recunoscută de gramatica "pozitivă": este corectă;
- Propoziţia este recunoscută folosind cel puţin o regulă "negativă": conţine cel puţin o greşeală gramaticală;
- Propoziţia nu este recunoscută: erorile de sintaxă sunt prea mari şi nu pot oferi nici o soluţie de corectare.

Pentru a creşte performanţele sistemului se pot introduce restricţii suplimentare în aplicarea regulilor negative; de exemplu se poate cere ca arborele Tr să nu aibă două reguli negative aplicate consecutiv.

Algoritmul *Grammar checker* va lucra astfel: la început se va încerca recunoaşterea secvenţei de intrare cu gramatica independentă de context "pozitivă" (folosind un algoritm de analiză oarecare - vezi [3]). Dacă ea reuşeşte, secvenţa este corectă; dacă nu, se reia procesul de analiză cu gramatica extinsă *NCFDG*, conform algoritmului 2 de sus. Dacă analiza reuşeşte, eroarea de sintaxă este găsită şi ea poate fi - eventual - corectată. Dacă nu, se reia Algoritmul 1 pentru a genera toate variantele posibile şi a le pune la dispoziţie utilizatorului pentru ca acesta să aibă o imagine de ansamblu a tuturor posibilităţilor de eroare.

Prelegerea 6

Gramatici pe arbori

6.1 Gramatici lexicalizate

Multe teorii lingvistice actuale acordă o importanţă crescută implementărilor lexicale în domenii de reprezentare considerate până acum pur sintactice. În particular, informaţia lexicală este incorporată în prelucrarea formalismelor gramaticale. Pentru detalii se poate face apel la [9]. În această prelegere vom defini numai formal un model de reprezentare lexical/sintactic a structurilor gramaticale specifice unei limbi: acest model, numit *Tree Adjoining Grammar* este folosit pe scară tot mai largă, cu aplicaţii diverse, de la structura lexicală până la analiza sintactică şi semantică a propoziţiilor. Pentru o informare succintă, a se vedea [6], [7], [8], [9].

Definiţia 6.1 *O gramatică* **lexicalizată** *este* $\mathcal{G} = (\mathcal{S}, \mathcal{A}, , f, < o_1, \ldots, o_n >)$ *unde:*

- $\mathcal{S}$ *(domeniul local de valabilitate) este format din o mulţime finită nevidă de elemente numite structuri;*
- $\mathcal{A}$ *este o mulţime finită nevidă de elemente numite termeni lexicali;*
- $f : \mathcal{S} \longrightarrow \mathcal{A}$ *este o aplicaţie care asociază fiecărei structuri câte un termen lexical; pentru* $\alpha \in \mathcal{S}$, $f(\alpha)$ *este numit* **ancora** *structurii* α.
- $o_i, 1 \leq i \leq n, n \geq 1$ *sunt operaţii de compunere a structurilor.*

Mulţimea $\mathcal{S}$ se numeşte **lexicon**. Structurile componente unui lexicon se numesc **structuri elementare**. Structurile obţinute folosind operaţii de compunere ale altor structuri se numesc **structuri derivate**.

Restricţii ale definiţiei:

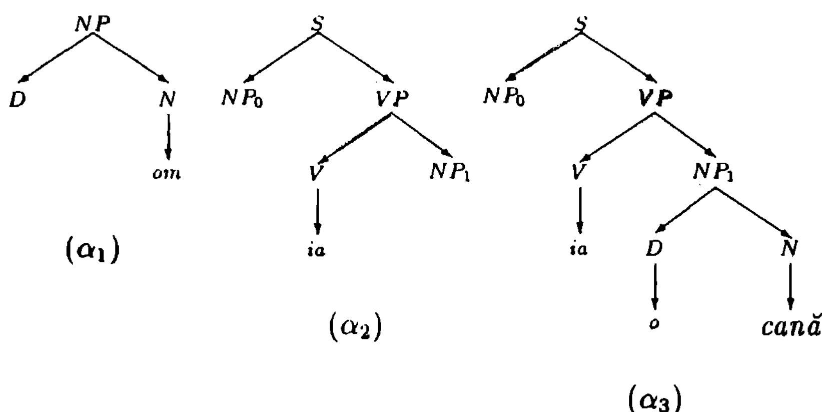
- Se vor considera doar structuri de mărime finită.
- Operaţiile de combinare vor avea atât ca operanzi cât şi ca rezultat un număr finit de structuri.
- Operaţiile nu vor putea copia, şterge sau rearanja (restructura) componentele nemărginite ale argumentelor lor.

Vom defini **categoria** unui cuvânt prin structura care îl selectează. Ea nu este redusă neapărat la o categorie singulară cum ar fi *verb*, *substantiv*, *adjectiv*, etc. Dacă o categorie are asociată o ancoră, spunem că întreaga structură defineşte categoria ancorei.

O structură este **lexicalizată** dacă în ea apare cel puţin un termen lexical deschis. Dacă apar mai mulţi termeni lexicali, atunci sau unul din ei este desemnat ca ancoră, sau întreaga mulţime de termeni lexicali locală structurii este desemnată ca o ancoră *multi-componentă*.

O gramatică compusă numai din structuri lexicalizate este evident lexicalizată.

Exemplul 6.1 *De exemplu, următoarele structuri sunt lexicalizate, conform definiţiei:*



Din definiţie rezultă imediat unele proprietăţi ale gramaticilor lexicalizate:

Lema 6.1 *Gramaticile lexicalizate sunt finite ambigui.*

Demonstraţie: O gramatică este finit ambiguă dacă nu există nici un cuvânt de lungime finită pentru care se pot construi un număr infinit de analize.

Afirmaţia din lemă este evidentă pentru propoziţii (secvenţe) de lungime arbitrară finită. Mulţimea structurilor ancorate de cuvintele acestor propoziţii constă din structurile necesare pentru analiza lor; oricare alte structuri introduc termeni lexicali care nu sunt în propoziţii.

Deoarece mulţimea structurilor selectate este finită, acestea pot fi combinate doar într-un număr finit de moduri. □

Pentru că orice propoziţie de lungime finită este finit ambiguă, spaţiul de lucru necesar analizei sintactice este finit. Deci problema de recunoaştere pentru gramaticile lexicalizate este decidabilă.

Lema 6.2 *Este decidabil dacă o propoziţie este acceptată sau nu de o gramatică lexicalizată.*

O consecinţă importantă a acestei leme este aceea că problema recunoaşterii pentru o unificare bazată pe gramatici lexicalizate este decidabilă; pentru gramatici nelexicalizate, ea este nedecidabilă.

Principala întrebare care apare acum este:

Pot fi lexicalizate gramaticile independente de context ?

Evident, nu orice *cfg* (gramatică independentă de context) este în formă lexicalizată. De exemplu, o regulă de tipul $S \longrightarrow NP\ VP$ nu este lexicalizată deoarece nu are în membrul drept nici un termen lexical.

Având o gramatică G definită într-un anumit formalism, să încercăm să găsim o gramatică echivalentă G_{lex} lexicalizată (nu neapărat în acelaşi formalism).

Definiţia 6.2 *Spunem că un formalism F poate fi lexicalizat de un alt formalism F' dacă pentru orice gramatică finit ambiguă G din F există o gramatică lexicalizată G' în F' cu proprietatea că G şi G' generează acelaşi set de arbori (şi deci acelaşi limbaj).*

6.2 Lexicalizarea cfg

După cum s-a văzut, o *cfg* lexicalizată are proprietatea că orice producţie conţine un simbol terminal în membrul drept; o astfel de regulă poate fi scrisă sub forma unei structuri cu o ancoră (corespunzătoare terminalului). Operaţiile de combinare sunt cele standard, rezultate din definiţia derivării.

O idee de a pune gramaticile independente de context în formă lexicalizată ar fi aducerea lor la *Forma Normală Greibach*. Reamintim, aceasta este o gramatică în care regulile sunt de forma $A \longrightarrow a\alpha, \alpha$ fiind o secvenţă de lungime maxim 2 formată numai din simboluri neterminale (a se vedea [2]).

Totuşi, aducerea unei gramatici la Forma Normală Greibach nu înseamnă lexicalizarea ei pentru că aceasta nu generează în general aceiaşi arbori de derivare cum sunt cei din gramatica originală. Deci această metodă nu poate oferi soluţia la cererea de lexicalizare.

În cele ce urmează vom lucra numai cu *cfg* de ambiguitate finită. Vom extinde domeniul lor local de valabilitate pentru a permite ca termenii lexicali să apară ca elemente ale structurilor elementare, folosind o gramatică pe arbori cu substituţia ca operaţie de combinare.

Această gramatică este formată dintr-un set de arbori care nu sunt condiţionaţi la adâncimea 1 (cum sunt producţiile unei gramatici independente de context), pe care se defineşte operaţia de substituţie (în loc de derivare directă). Substituţiile pot înlocui un nod neterminal aflat pe frontiera unui arbore. Prin convenţie, nodurile care se pot înlocui (substitui) sunt marcate cu $\downarrow$. După substituirea unui nod n, acesta este înlocuit cu subarborele respectiv.

Definiţia 6.3 *O gramatică de substituţie pe arbori (TSG - Tree Substitution Grammar) este o structură $G = (V_N, V_T, I, S)$, unde:*

1. *V_T este o mulţime finită nevidă de simboluri terminale;*
2. *V_N este o mulţime finită nevidă de simboluri neterminale, $V_T \cap V_N = \emptyset$;*
3. *$S \in V_N$ este un simbol special;*
4. *I este o mulţime finită nevidă de arbori finiţi ale căror noduri interioare sunt marcate cu elemente din V_N iar cele de pe frontieră sunt marcate cu elemente din $V_N \cup V_T$. Toate nodurile neterminale de pe frontieră sunt marcate pentru substituţie cu $\downarrow$.*

Spunem că un arbore este *derivat* dacă s-a obţinut dintr-un arbore iniţial în care s-au efectuat substituţii cu arbori iniţiali sau derivaţi. Un X-arbore este un arbore a cărui rădăcină este marcată cu X.

Un arbore este *complet* dacă frontiera lui este formată numai din noduri marcate cu simboluri terminale.

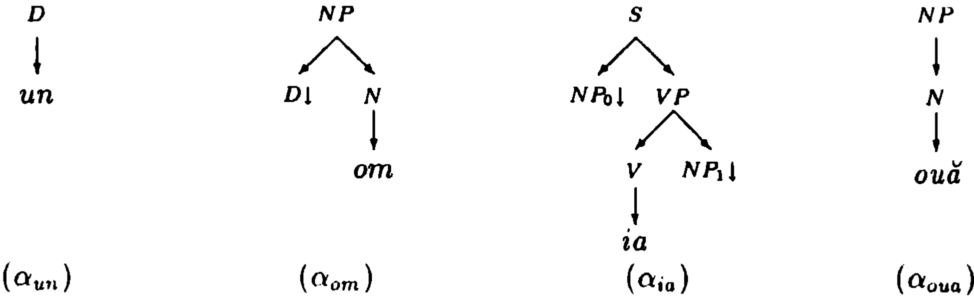
Vom nota frontiera $\alpha \in V_N \cup V_T$ a unui X-arbore t_X cu $Fr(t_X) = \alpha$.

Fiind daţi doi arbori t_A, t_B, definim derivarea $t_A \Longrightarrow t_B$ astfel:

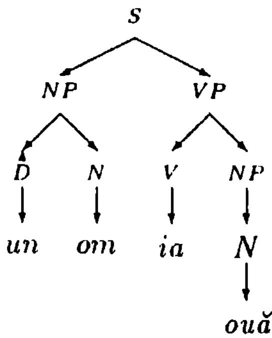
- t_A este un A-arbore, $Fr(t_A) = w$;
- Există un B-arbore iniţial t cu $Fr(t) = uA\downarrow v$;
- t_B este un B arbore cu $Fr(t_B) = uwv$ obţinut din t prin înlocuirea nodului marcat $A\downarrow$ cu t_A.

Observaţie: t_A poate fi arbore iniţial sau derivat, iar t_B este un arbore derivat.

Exemplul 6.2 *Din arborii iniţiali*



se obţine arborele derivat



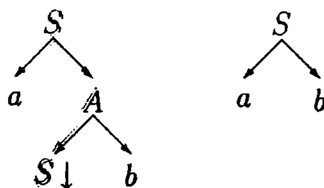
Fie $\Longrightarrow^*$ închiderea reflexivă şi tranzitivă a relaţiei $\Longrightarrow$. Definim limbajul generat de o TSG G_T prin

$$L(G_T) = \{t | t \text{ este } S - \text{arbore } Fr(t) \in V_T^*,\ \exists\, t' \in I,\ t' \Longrightarrow^* t\}$$

Spunem că o cfg G este echivalentă cu o TSG G_T dacă:

- Orice arbore de derivare din G este în $L(G_T)$;
- $\forall\, t \in L(G_T)$, $\exists\, \alpha \in L(G)$ cu $Fr(t) = \alpha$.

Exemplul 6.3 *Gramatica definită prin arborii iniţiali*



este echivalentă cu gramatica de producţii

$$S \longrightarrow aA|ab, \quad A \longrightarrow Sb$$

care generează limbajul $L(G) = \{a^n b^n | n \geq 1\}$.

Este uşor de demonstrat că mulţimea limbajelor generate de gramaticile de substituţie pe arbori coincide cu $\mathcal{L}_2$ (clasa limbajelor independente de context).

Să revenim la problema lexicalizării gramaticilor independente de context. Un prim rezultat este:

Lema 6.3 *Gramaticile independente de context finit ambigui nu pot fi lexicalizate cu gramatici de substituţie pe arbori.*

Demonstraţie: Să presupunem prin absurd că acest lucru este posibil şi să considerăm gramatica de producţii:

$$S \longrightarrow SS, \quad S \longrightarrow a.$$

Fie G gramatica lexicalizată echivalentă şi să luă un arbore iniţial arbitrar t. Din definiţie, există un nod n pe frontiera lui t marcat cu a. Pentru că substituţiile pot fi efectuate numai pe frontiera arborilor iniţiali (conform derivărilor in G), distanţa dintre n şi rădăcina lui t este constantă pentru orice arbore iniţial. Deci, în orice arbore derivat din G există o ramură de lungime fixată de la rădăcină la un nod notat a, ramură ce nu mai poate fi extinsă. Dar în cfg definită mai sus se pot construi arbori de derivare în care orice drum la un nod terminal are o lungime oricât de mare. Contradicţie. ◻

6.3 Gramatici *TAG*

Rezultatul prezentat anterior conduce la ideea de a mări puterea generativă a gramaticilor pe arbori prin introducerea unei operaţii noi, care să insereze arbori în arbori. Această operaţie, numită **adjuncţie** a fost definită în [6] şi [7] şi a condus la o nouă clasă de sisteme generative, *TAG*-urile.

Definiţia 6.4 *O gramatică TAG (Tree Adjoining Grammar) este un quintuplu* (V_N, V_T, I, A, S), *unde:*

1. V_T *este o mulţime finită nevidă de simboluri terminale;*
2. V_N *este o mulţime finită nevidă de simboluri neterminale,* $V_T \cap V_T = \emptyset$;
3. $S \in V_N$ *este simbolul de start;*
4. *I este o mulţime finită nevidă de arbori finiţi numiţi arbori iniţiali şi caracterizaţi prin:*
   - *nodurile interioare sunt etichetate cu simboluri neterminale;*
   - *nodurile de pe frontieră sunt etichetate cu terminale sau neterminale;*
   - *simbolurile neterminale de pe frontieră sunt marcate pentru substituţie cu* $\downarrow$;
5. *A este o mulţime finită nevidă de arbori finiţi numiţi arbori auxiliari şi caracterizaţi prin:*
   - *nodurile interioare sunt etichetate cu simboluri neterminale;*
   - *nodurile de pe frontieră sunt etichetate cu simboluri terminale sau neterminale; neterminalele sunt marcate pentru substituţie cu* $\downarrow$ *înafară de un singur nod numit "picior", care este marcat cu asterisc* $(*)$;
   - *eticheta piciorului coincide cu eticheta nodului rădăcină al arborelui.*

În gramaticile *TAG* lexicalizate, fiecare arbore din $I \cup A$ are cel puţin un simbol terminal (ancora) pe frontieră.

Arborii din $I \cup A$ sunt numiţi *arbori elementari.* Un X-arbore elementar este un arbore din $I \cup A$ care are neterminalul X ca etichetă a rădăcinii.

Un arbore construit prin compunerea a doi arbori este numit *arbore derivat.* Când un nod este marcat pentru substituţie, numai arborii derivaţi din arborii iniţiali pot fi substituiţi în poziţia lui.

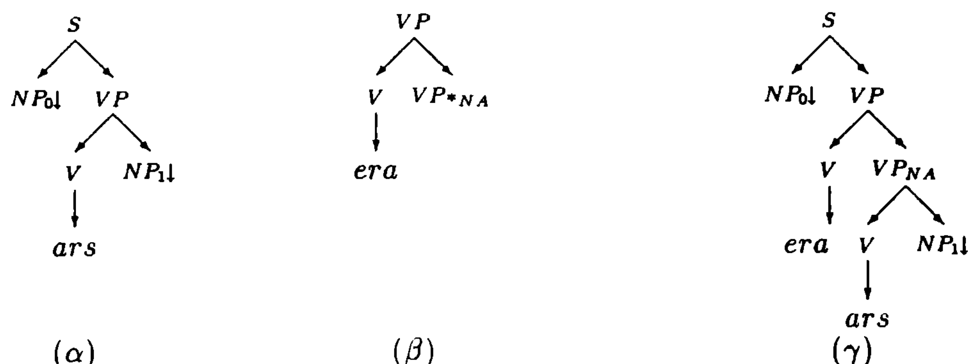
Să definim acum operaţia de **adjuncţie**. Fie α un arbore (iniţial, auxiliar sau derivat) care are un nod n etichetat cu X şi nemarcat (pentru substituţie); considerăm β un X-arbore auxiliar. Arborele γ rezultat prin adjuncţia lui β la α în nodul n se construieşte astfel:

1. Subarborele t al lui α dominat de n este şters din α, păstrându-se o copie a sa;

2. Arborele β este ataşat nodului n;

3. Copia lui t este ataşată piciorului lui β (inserat în α).

Exemplul 6.4 *Un exemplu simplu de folosire a operaţiei de ajuncţie este:*



Din considerente lingvistice apare necesitatea de a construi şi o precizare a arborilor auxiliari care pot fi adjuncţi unui nod dat (ceva similar dependenţei de context în gramaticile Chomski). Aceasta se realizează prin *restricţiile unei adjuncţii* ([7],[8])

Fie $G = (V_N, V_T, I, A, S)$ un TAG. Pentru orice nod al unui arbore elementar se pot defini trei tipuri de restricţii ale adjuncţiilor:

1. **Adjuncţie selectivă** ($SA(T)$) - arborii care pot fi adjunctaţi nodului sunt din mulţimea $T \subseteq A$; adjuncţia nu este obligatorie.

2. **Adjuncţie nulă** (NA) - nodul respectiv nu este implicat în nici o operaţie de adjuncţie; restricţia este echivalentă cu $SA(\emptyset)$.

3. **Adjuncţie obligatorie** ($OA(T)$) - nodul trebuie să fie modificat cu o adjuncţie folosind un arbore din $T \subseteq A$. Pentru $OA(A)$ se foloseşte prescurtarea OA.

În Exemplul 6.4, nodul VP din arborele β are restricţia NA, ceea ce înseamnă că la el nu se poate face nici o adjuncţie.

Dacă nu se fac alte specificaţii, vom folosi în continuare gramatici TAG cu substituţii, adjuncţii şi restricţii pe adjuncţii.

Pentru un TAG $G = (V_N, V_T, I, A, S)$, mulţimea arborilor $T(G)$ este mulţimea tuturor S-arborilor derivaţi t cu $Fr(t) \in V_T^*$. Aceşti arbori sunt obţinuţi plecând de la arborii iniţiali.

Se mai poate defini şi limbajul generat de un G ca fiind

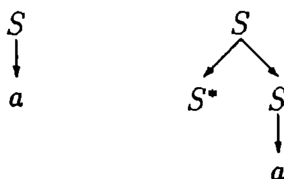
$$L(G) = \{Fr(t) | t \in T(G)\}.$$

Similar gramaticilor independente de context, se poate defini şi aici arborele de derivare; interpretarea sa este însă puţin diferită: un arbore de derivare este un obiect care specifică în mod unic construcţia arborelui derivat. Rădăcina sa este S (deci este un S-arbore), nodurile sale sunt rădăcini pentru arbori auxiliari (pentru adjuncţie) sau iniţiali (pentru substituţie), ordinea interpretării derivării în arbore este arbitrară.

Exemplul 6.5 *Gramatica de producţii*

$$S \longrightarrow SS, \quad S \longrightarrow a$$

care nu putea fi lexicalizată cu un TSG, poate fi lexicalizată folosind adjuncţia, astfel:



De remarcat că a este ancora ambilor arbori (cel iniţial şi cel auxiliar).

Teorema 6.1 *Dacă $G = (V_N, V_T, S, P)$ este o cfg finit ambiguă cu $\epsilon \notin L(G)$, atunci există o TAG lexicalizată $G_{lex} = (V_N, V_T, I, A, S)$ care generează acelaşi limbaj şi arbori ca şi G. Mai mult, G_{lex} poate fi definită să nu aibă noduri de sunstituţie.*

Demonstraţie: Fie $G = (V_N, V_T, S, P)$ o *cfg* finit ambiguă cu $\epsilon \notin L(G)$. Spunem că $A \in V_N$ este un simbol recursiv dacă şi numai dacă $\exists \alpha, \beta \in (V_N \cup V_T)^*, A \Longrightarrow^* \alpha A \beta$. O producţie $A \longrightarrow \delta$ este recursivă dacă A este recursiv. Se pot da diverşi algoritmi pentru determinarea mulţimilor simbolurilor (regulilor) recursive şi nerecursive ([2]).

Partiţionăm mulţimea P a producţiilor lui G în două: mulţimea R a regulilor recursive şi mulţimea NR a regulilor nerecursive. Evident $R \cup NR = V_N, R \cap NR = \emptyset$.

Fie $L(NR) = \{w | S \Longrightarrow^* w\}$ unde în derivări s-au folosit numai producţii din NR. Evident, $L(NR)$ este finită şi $\epsilon \notin L(NR)$. Fie I mulţimea arborilor de derivare definiţi de $L(NR)$. I este o mulţime finită şi, toti arborii din I au pe frontieră cel puţin un simbol terminal, deoarece $\epsilon \notin L(NR)$. I va fi mulţimea arborilor iniţiali din gramatica lexicalizată G_{lex}.

Să formăm acum o bază de cicluri minime. Se pot folosi algoritmi clasici de grafuri pentru a găsi o mulţime finită de *cicluri de bază* astfel încât:

- ele nu au subcicluri;
- orice ciclu este o combinaţie de astfel de cicluri.

Fie $\{c_1, \ldots, c_k\}$ o bază de cicluri minime. Aplicăm următorul algoritm pentru construcţia mulţimii arborilor auxiliari.

1. $A := \emptyset$

2. pentru toţi $c_i, 1 \leq i \leq k$

   pentru toate nodurile n_j din c_i cu eticheta B_j

   fie $B_j \Longrightarrow^* \alpha_j B_j \beta_j$ (conform cu c_i);

   dacă B_j este eticheta unui nod dintr-un arbore din $I \cup A$ atunci

   pentru toate derivările $\alpha_j \Longrightarrow^* w_j \in V_T^*$, $\beta_j \Longrightarrow^* z_j \in V_T^*$

   care folosesc numai reguli nerecursive,

   se adaugă la A arborii auxiliari corspunzători derivării

   $B_j \Longrightarrow^* \alpha_j B_j \beta_j \Longrightarrow^* w_j B_j z_j$, unde nodul B_j de pe frontieră

   este picior.

Construcţia de mai sus asigură că toţi arborii auxiliari au cel puţin un termen lexical pe frontieră ($\alpha_j \beta_j$ trebuie să ducă într-o secvenţă terminală, altfel se ajunge la derivarea $B_j \Longrightarrow^* B_j$, imposibilă pentru o gramatică finit ambiguă).

G_{lex} astfel construită este lexicalizată şi generează acelaşi set de arbori ca şi G. Teorema este demonstrată. □

Evident, construcţia de mai sus nu conduce la o gramatică optimă din punct de vedere al numărului de elemente din $I \cup A$; dar nu acest lucru s-a avut în vedere.

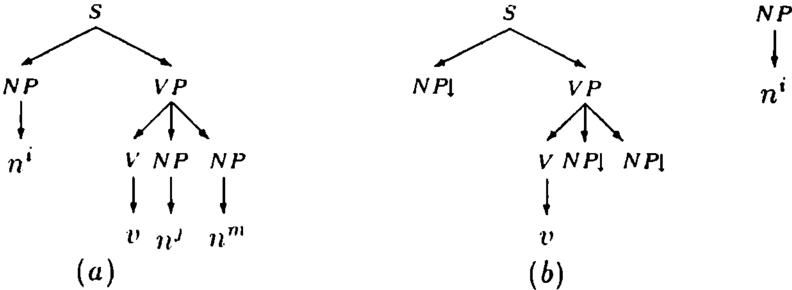
O altă concluzie a demonstraţiei anterioare este aceea că adjuncţia este suficientă pentru lexicalizarea gramaticilor independente de context. Totuşi, luând adjuncţia ca bază, substituţia este utilă pentru a scădea gradul de complexitate al gramaticilor. Să dăm un exemplu pentru a ilustra această afirmaţie:

Exemplul 6.6 *Fie gramatica de producţii*

$$S \longrightarrow NP\ VP, \quad VP \longrightarrow V\ NP\ NP, \quad V \longrightarrow v, \quad NP \longrightarrow n^i; i \in [1, k].$$

unde S, NP, VP, $V \in V_N$, $n, v \in V_T$.

În această gramatică se pot genera k^3 *arbori de derivare de tipul* (a) $(i, j, m \in [1, k])$*:*



Deoarece gramatica nu are producţii recursive, nu vor fi derivări de forma $A \Longrightarrow^* \alpha A \beta$. *Deci, orice TAG care lexicalizează această gramatică nu va avea arbori auxiliari (deoarece la un astfel de arbore rădăcina şi piciorul au acelaşi neterminal ca etichetă). Rezultă că dacă se foloseşte numai adjuncţia, toate derivările* (k^3)

reprezintă arbori iniţiali în gramatica lexicalizată. Dacă se foloseşte şi substituţia, sunt suficienţi $k+1$ *arbori iniţiali (vezi (b)).*

Deci, rata de compactare este $O(n^2)$.

În general, pentru gramatici de acest tip, folosirea substituţiei ca o a doua operaţie scade numărul de arbori iniţiali de la $O(k^3)$ la $O(k)$.

Se poate arăta ca o proprietate remarcabilă, faptul că mulţimea gramaticilor TAG este închisă la *operaţia* de lexicalizare.

Teorema 6.2 *Dacă G este un TAG finit ambiguu* ($\epsilon \notin L(G)$) *cu substituţia şi adjuncţia ca operaţii, atunci există un TAG lexicalizat* G_{lex} *care generează acelaşi limbaj şi aceiaşi arbori ca G.*

Demonstraţie: Demonstraţia este similară celei de la teorema anterioară; de aceea o vom schiţa doar. Separăm din nou partea recursivă a gramaticii de cea nerecursivă. Partea recursivă formează arborii auxiliari din G_{lex}; deoarece G este finit ambiguă, toţi arborii auxiliari vor avea cel puţin un simbol terminal pe frontieră. Producţiile nerecursive vor genera arborii iniţiali. Cum $\epsilon \notin L(G)$, şi aceştia vor avea cel puţin un simbol terminal pe frontieră. □

Pentru aplicarea TAG-urilor în construcţii lingvistice se poate folosi (în cazul limbii engleze) [9].

Prelegerea 7

Extensii ale cfg-urilor

După cum se ştie, majoritatea analizorilor sintactici se bazează pe gramatici independente de context; acestea sunt însă prea restrictive şi nu pot cuprinde toată bogăţia unui limbaj natural. De aceea vom face o extensie a acestor gramatici asociindu-le un set de caracteristici.

7.1 Sisteme de caracteristici şi gramatici augmentate

În limbajele naturale există noţiunea de acord între cuvinte în cadrul unei propoziţii. De exemplu, construcţia *NP*: *un creioane* nu este corectă, deoarece articolul nehotărât *un* este la singular, iar substantivul *creioane* este la plural. Spunem că acest *NP* nu satisface **acordul de număr** al limbii române. Există multe alte acorduri, cum ar fi acordul *subiect-predicat*, acordul de *gen pentru pronume*, şi altele. Pentru a controla aceste fenomene de limbă, vom extinde formalismul gramatical prin adăugarea de constituienţi numiţi **caracteristici**.

De exemplu, putem defini caracteristica *NUMĂR* care poate lua două valori: *s* (pentru singular) respectiv *p* (pentru plural); pe baza ei, o regulă poate fi:

$$NP \longrightarrow ART\ N \qquad \text{doar dacă } NUMĂR1 \text{ este în acord cu } NUMĂR2$$

Semnificaţia ei este: *o expresie substantivală corectă este formată dintr-un articol urmat de un substantiv, cu condiţia ca cele două cuvinte să fie în acord relativ la număr.*

Această producţie este echivalentă cu două reguli independente de context care vor folosi terminale distincte pentru codificarea formei singular şi respectiv plural ale expresiilor substantivale:

$$NP._{SING} \longrightarrow ART._{SING}\ N._{SING}$$
$$NP._{PLURAL} \longrightarrow ART._{PLURAL}\ N._{PLURAL}$$

Utilizarea unei astfel de construcţii ar conduce însă la dublarea numărului de producţii din gramatică. Şi în mod similar, orice caracteristică nouă ar multiplica numărul producţiilor în mod artificial. Din acest punct de vedere, este preferabilă

maniera de definire formală a caracteristicilor (şi deci dimensiunea gramaticii va rămâne neschimbată).

Pentru a realiza aceasta, se definesc constituienţi numiţi ***structuri caracteristice***; ei sunt de fapt aplicaţii parţial definite de la caracteristici la valori de caracteristici, care definesc proprietăţi relevante ale constituienţilor.

De exemplu, o structură caracteristică a unui constituient *ART1* care reprezintă o anumită utilizare a unui cuvânt α poate fi scrisă astfel:

ART1 : (CAT ART
ROOT α
NUMĂR s)

Aceasta spune că *ART1* este un constituient din categoria *ART* care are ca rădăcină cuvântul α şi este la numărul singular.

De obicei se foloseşte o altă scriere, care se apropie de modul de reprezentare al producţiilor unei gramatici independente de context. Astfel, definiţia de mai sus are forma:

ART1:(ART ROOT α *NUMĂR s)*

Structurile caracteristice pot fi folosite ele însele ca valori, ducând la formarea de constituienţi mai complexi. Într-o astfel de construcţie, ordinea de apariţie a substructurilor este numerotată. Astfel, pentru regula

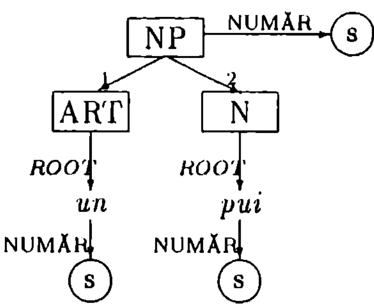
NP ⟶ *ART N*

reprezentarea de exemplu a expresiei substantivale *NP: un pui*, poate fi:

*NP*1	:(*NP*	NUMĂR s	
	1	(ART	ROOT *un*
			NUMĂR s)
	2	(N	ROOT *pui*
			NUMĂR s))

Acelaşi constituient se poate reprezenta sub formă arborescentă astfel:

Figura 7.1:



Pentru a surprinde aceste caracteristici, vom folosi o ***augmentare*** a producţiilor gramaticii, în care neterminalele sunt valori de caracteristici.

De exemplu, pentru regula folosită anterior, producţia augmentată este:

(*NP* NUMĂR ?*n*) ⟶ (*ART* NUMĂR ?*n*)(*N* NUMĂR ?*n*)

Semnificaţia ei este: un constituient *NP* constă din doi subconstituienţi, primul fiind un *ART* iar al doilea un *N*, în care caracteristica *NUMĂR* este identică la toţi trei. Conform acestei reguli, constituientul *NP1* dat mai sus este corect. Pe de-altă parte, constituientul

(*NP* 1 (ART NUMĂR s)
  2 (N NUMĂR s))

nu este corect, deoarece *NP* nu are definită caracteristica *NUMĂR*; la-fel,

(*NP* NUMĂR s
  1 (ART NUMĂR s)
  2 (N NUMĂR p))

nu este corect deoarece caracteristica *NUMĂR* nu are aceeaşi valoare peste tot.

Prin această definiţie se pot rezolva şi problemele de ambiguitate dintr-un constituient. De exemplu, cuvântul *pui* (considerat substantiv) este ambiguu deoarece are aceeaşi formă pentru singular şi pentru plural. Deci, două din valorile sale din lexicon vor diferi numai prin valoarea caracteristicii *NUMĂR*. Putem simplifica aceasta definind o singură intrare care foloseşte o variabilă drept valoare a acestei caracteristici; deci:

(N ROOT *pui* NUMĂR ?n)

Aceasta înseamnă că orice valoare a caracteristicii *NUMĂR* este corectă pentru cuvântul *pui*. În multe cazuri însă, termenul de *"orice valoare"* trebuie înlocuit cu acela de *"domeniu de valori"*; sunt introduse astfel **variabile restrictive** - variabile care pot lua valori numai dintr-o mulţime specificată. De exemplu,

$$?n\{s,p\}$$

este o variabilă care poate lua numai valorile s sau p. Adesea, în astfel de definiţii, numele variabilei devine inutil şi poate fi eliminat. Astfel, putem scrie

(N ROOT *pui* NUMĂR ?n{s,p})

dar şi - mai simplu

(N ROOT *pui* NUMĂR {s.p})

O problemă teoretică care se pune este aceea dacă puterea generativă a gramaticilor astfel augmentate nu depăşeşte cumva $\mathcal{L}_2$. Este clar că - în cazul în care mulţimile de valori pentru restricţiile caracteristicilor sunt finite, se pot crea categorii de constituienţi pentru toate combinaţiile de caracteristici, ceea ce conduce tot la o gramatică independentă de context. Pentru mulţimi arbitrare de valori, puterea generativă o egalează pe cea a maşinilor Turing. Totuşi, din punct de vedere practic, chiar dacă o mulţime de valori nu este restrânsă explicit, ea nu parcurge decât o mulţime finită, deci se pot folosi algoritmii standard de analiză construiţi pentru gramaticile independente de context.

7.2 Câteva sisteme de caracteristici

7.2.1 Caracteristici de număr şi persoană

Aceste două restricţii - numite şi *acord număr - persoană* - sunt foarte des folosite în gramatica limbii române. Locul unde ele sunt întâlnite cel mai frecvent este la îmbinarea subiect - predicat.

Deoarece - aşa cum se ştie din practică - ele sunt dependente unul de altul în conjugări, este convenabil să se creeze o caracteristică comună *AGR* - cu şase valori posibile (persoanele I,II,III singular şi plural) codificate respectiv prin *1s*, *2s*, *3s*, *1p*, *2p*, *3p*.

De exemplu, cuvântul *este* poate fi corect numai pentru persoana a III-a singular; deci caracteristica sa *AGR* este *3s*. Pentru cuvântul *scriu*, caracteristica *AGR* este o valoare din mulţimea {*1s*, *3p*}.

7.2.2 Caracteristici ale verbelor

Există o caracteristică importantă a limbii - *VFORM* - legată de forma verbelor în conjugare. Utilitatea ei apare mai ales la analiza părţilor auxiliare de propoziţie şi la stabilirea de restricţii de împărţire pe subdomenii pentru prefixe a verbelor. Limba română prezintă destule complicaţii; de aceea, pentru a surprinde anumite nuanţe, vom lucra cu următoarele valori ale caracteristicii *VFORM*:

Tabelul 7.1:

inf	:	infinitiv: *a fi*, *a lucra*, *a şti*, *a culege*, ...
ind	:	indicativ: *fi*, *lucra*, *şti*, *culege*, ...
pres	:	indicativ prezent (sau prezentul simplu): *sunt*, *lucrez*, *ştiu*, *culeg*, ...
imp	:	indicativ trecut - imperfect: *eram*, *lucram*, *ştiam*, *culegeam*, ...
sim	:	indicativ trecut - perfectul simplu: *fusei*, *lucrai*, *ştiui*, *culesei*, ...
conj	:	conjunctiv: *să fiu*, *să lucrez*, *să ştiu*, *să culeg*, ...
prt	:	participiu: *fost*, *lucrat*, *ştiut*, *cules*, ...
ger	:	gerunziu: *fiind*, *lucrând*, *ştiind*, *culegând*, ...

Pentru a putea realiza interacţiunea dintre predicate şi complementele lor, se foloseşte o altă caracteristică, *SUBCAT*. În Tabelul 7.2 sunt prezentate câteva valori ale acestei caracteristici pentru complemente formate din combinaţii $NP - VP$.

Să luăm din acest tabel de exemplu valoarea *_np_vp:conj*; ea va fi utilizată pentru a indica un complement format dintr-un *NP* urmat de un VP a cărui valoare caracteristică *VFORM* este *conj*. Această regulă poate fi scrisă sub forma unei producţii astfel:

$$
\begin{aligned}
(VP) \longrightarrow\ & (V\ SUBCAT\ _np_vp : conj) \\
& (NP) \\
& (VP\ VFORM\ conj)
\end{aligned}
$$

Explicitând în cuvinte, o expresie verbală *VP* constă dintr-un *V* cu valoarea *SUBCAT* *_np_vp:conj*, urmat de un *NP*, urmat de un *VP* cu valoarea *VFORM* *conj*.

Tabelul 7.2:

Valoare	Exemplu verb	Exemplu propoziţie
_none	*a râde*	Maria a râs.
_np	*a găsi*	Maria a găsit o cheie.
_np_np	*a da*	Paula îi dă Danei notiţele.
_vp : conj	*a vrea*	Ion vrea să zboare.
_np_vp : conj	*a spune*	Radu îi spune Anei să plece.
_vp : ger	*a plânge*	Petre plânge râzând.

Multe verbe sunt însoţite de alte tipuri de complemente (de loc, de timp, etc). Pentru acestea există expresii prepoziţionale (în particular prepoziţii) specifice. Astfel, verbul *a da* poate primi un complement format dintr-un *NP* urmat de o *PP* care foloseşte prepoziţia *pentru*, ca în:

Andrei dă bani pentru maşină.

Alte verbe (*a pune, a se duce*) necesită expresii prepoziţionale care exprimă un loc, şi folosesc prepoziţii cum ar fi *din, de la, în*, ...

O caracteristică care să cuprindă expresiile prepoziţionale este *PFORM*. O *PP* cu o valoare *PFORM* α trebuie să înceapă cu α. Se pot face diverse clasificări ale valorilor acestei caracteristici. Printre cele mai semnificative valori sunt:

- Valori din familia *PENTRU* (prepoziţii de tipul *pentru, ca să, contra*,...;
- Valori din familia *LOC*: sunt prepoziţii care desemnează un complement de loc.
- Valori din familia *MOT*, sunt folosite pentru anumite aspecte ale mişcării. *MOT* este asociată verbelor de tipul *a merge* şi conţine prepoziţii cum ar fi *la, din, de-a lungul, printre*,....

Tabelul 7.3:

Valoare	Exemplu prepoziţie	Exemplu propoziţie
PENTRU	*pentru, contra*	L-am luat pentru totdeauna.
LOC	*în, pe, prin*	Am pus-o pe masă.
MOT	*spre, de la*	A luat-o spre gară.

Ultimele două tipuri de valori par a nu fi distincte la prima vedere (de exemplu *la* aparţine ambelor valori); încercând cu alte prepoziţii din acelaşi set, distincţia se face imediat. De exemplu,

Petre pune hârtia la coş

este o construcţie corectă, dar

Petre pune hârtia printre coş,
este greşită. Deci, în acest caz, prepoziţia *la* aparţine valorii *PFORM LOC.*
În Tabelul 7.3 sunt date câteva exemple de valori ale caracteristicii *PFORM.*

Această caracteristică poate fi folosită pentru a sistematiza formele de complement ale verbelor. Astfel, valoarea *SUBCAT* a unui verb cum ar fi ***a pune*** este *_np_pp:loc*, iar o regulă scrisă similar producţiilor gramaticale va avea forma:

$$(VP) \longrightarrow \begin{array}{l}(V\ SUBCAT\ _np_pp : loc)\\ (NP)\\ (PP\ PFORM\ LOC)\end{array}$$

În cazul propoziţiilor dependente unele de altele, noţiunea de completivă apare în mod natural şi va trebui să construim o subcategorie pentru ea; o vom numi caracteristica *COMP*, cu valori posibile *ca, pentru, că, pentru că,...* De exemplu, verbul *a spune* poate solicita o completivă prin *că*; deci, o valoare *SUBCAT* pentru *a spune* va fi *_s:că.*

In Tabelul 7.4 sunt listate câteva valori SUBCAT suplimentare şi exemple pentru câteva verbe.

Tabelul 7.4:

Valoare	**Exemplu verb**	**Exemplu propoziţie**
_np_pp : lui	*a da*	Ion a dat un pix lui Marin.
_pp : loc	*a fi*	Adrian este la şcoală.
_np_pp : loc	*a pune*	Eugen a pus cutia în colţ.
_pp : mot	*a se duce*	Laura s-a dus la magazin.
_np_pp : mot	*a lua*	Ana a luat copilul la piaţă.
_adjp	*a fi*	Paul este fericit.
_np_adjp	*a ţine*	Soţia ţine cina caldă.
_s:că	*a crede*	Radu crede că lumea e mică.
_s:pentru că	*a se ruga*	Ana se roagă pentru că este creştină.

7.2.3 Caracteristici binare

Anumite caracteristici au proprietatea că orice constituient posedă sau nu caracteristica respectivă (deci o decizie binară). Putem formaliza aceasta ca o caracteristică cu mulţimea valorilor $\{-,+\}$.

De exemplu, caracteristica *INV* indică dacă o propoziţie *S* este enunţiativă sau interogativă. Deci, *S* - structura pentru propoziţia

Ion râde.

are o *INV* valoare $-$, în timp ce *S* - structura pentru propoziţia

Ion râde ?

are valoarea *INV* $+$.

Adesea această valoare este folosită ca prefix, şi se scrie $+$ *INV* sau $-$ *INV*.

7.2.4 Valoarea "eroare" pentru caracteristici

În multe ocazii este util să alocăm pentru caracteristici şi o valoare de eroare. Ori de câte ori un constituient este construit să aibă o caracteristică iar valoarea acelei caracteristici lipseşte (nu este specificată), caracteristica va lua valoarea de eroare –.

Această valoare este inserată automat la prima construire a constituientului.

7.3 Analiza morfologică şi lexiconul

Înainte de a ne pune problema construirii unei gramatici, va trebui definit lexiconul pe care se bazează limbajul generat de gramatică. Acest lexicon va conţine (după cum am văzut în exemplele anterioare) informaţii despre toate cuvintele diferite care pot fi folosite, inclusiv valorile tuturor caracteristicilor. Dacă un cuvânt este ambiguu, el trebuie descris prin mai multe poziţii în lexicon, câte una pentru fiecare variantă.

Deoarece cuvintele au tendinţa de a se supune unor tipuri morfologice regulate, multe forme ale lor nu vor trebui introduse explicit în lexicon. Conjugările multor verbe de exemplu, folosesc aceeaşi formă la persoana a treia singular şi plural, formează infinitivul cu prefixul *a* şi gerunziul cu sufixul *-ind* sau *-înd*. Aceste elemente reduc mult numărul de intrări în lexicon.

Tabelul 7.5:

	Indicativ prezent:
1.	(*V ROOT ?r SUBCAT ?s VFORM prez AGR* 3*s*) ⟶ (*V ROOT ?r SUBCAT ?s VFORM ind IRREG – PRES*–)(+*S*)
2.	(*V ROOT ?r SUBCAT ?s VFORM prez AGR* (1*s* 2*s* 1*p* 2*p* 3*p*)) ⟶ (*V ROOT ?r SUBCAT ?s VFORM ind IRREG – PRES*–)
	Indicativ trecut:
3.	(*V ROOT ?r SUBCAT ?s VFORM past AGR* (1*s* 2*s* 1*p* 2*p* 3*p*)) ⟶ (*V ROOT ?r SUBCAT ?s VFORM ind IRREG – PAST*–)(+*ED*)
	Participiu:
4.	(*V ROOT ?r SUBCAT ?s VFORM pastprt*) ⟶ (*V ROOT ?r SUBCAT ?s VFORM ind ED – PASTPRT*–)(+*ED*)
5.	(*V ROOT ?r SUBCAT ?s VFORM pastprt*) ⟶ (*V ROOT ?r SUBCAT ?s VFORM ind EN – PASTPRT*–)(+*EN*)
	Gerunziu:
6.	(*V ROOT ?r SUBCAT ?s VFORM ing*) ⟶ (*VROOT ?r SUBCAT ?s VFORM ind*)(+*ING*)
	Pluralul substantivelor:
7.	(*N ROOT ?r AGR* 3*p*) ⟶ (*N ROOT ?r AGR* 3*s IRREG – PL*–)(+*S*)

Tabelul 7.6:

a	$(CAT\ ART\ ROOT\ A1\ AGR\ 3s)$
be	$(CAT\ V\ ROOT\ BE1\ VFORM\ ind\ IRREG-PRES+\ IRREG-PAST+$ $SUBCAT\ \{_adjp_np\})$
cry	$(CAT\ V\ ROOT\ CRY1\ VFORM\ ind\ SUBCAT\ _none)$
dog	$(CAT\ N\ ROOT\ DOG1\ AGR\ 3s)$
fish	$(CAT\ N\ ROOT\ FISH1\ AGR\ \{3s\ 3p\})$
happy	$(CAT\ ADJ\ SUBCAT\ _vp:inf)$
he	$(CAT\ PRO\ ROOT\ HE1\ AGR\ 3s)$
is	$(CAT\ V\ ROOT\ BE1\ VFORM\ prez\ SUBCAT\ \{_adjp_np\}\ AGR\ 3s)$
Jack	$(CAT\ Nume\ AGR\ 3s)$
man	$(CAT\ N\ ROOT\ MAN1\ AGR\ 3s)$
men	$(CAT\ N\ ROOT\ MAN1\ AGR\ 3p)$
saw	$(CAT\ N\ ROOT\ SAW1\ AGR\ 3s)$
saw	$(CAT\ V\ ROOT\ SAW2\ VFORM\ ind\ SUBCAT\ _np)$
saw	$(CAT\ V\ ROOT\ SEE2\ VFORM\ past\ SUBCAT\ _np)$
see	$(CAT\ V\ ROOT\ SEE1\ VFORM\ ind\ SUBCAT\ _np$ $IRREG-PAST+\ EN-PASTPRT+)$
seed	$(CAT\ N\ ROOT\ SEED1\ AGR\ 3s)$
the	$(CAT\ ART\ ROOT\ THE1\ AGR\ \{3s\ 3p\})$
to	$(CAT\ TO)$
want	$(V\ ROOT\ WANT1\ VFORM\ ind\ SUBCAT\ \{_np_vp:inf\ _np_vp:ger\})$
was	$(CAT\ V\ ROOT\ BE1\ VFORM\ past\ AGR\ \{1s\ 3s\}\ SUBCAT\ \{_adjp_np\})$
were	$(CAT\ V\ ROOT\ BE1\ VFORM\ past\ AGR\ \{2s\ 1p\ 2p\ 3p\}$ $SUBCAT\ \{_adjp_np\})$

Datorită numărului mare de variante prezentate de gramatica şi lexiconul limbii române, în această parte vom folosi exemple preluate din limba engleză, unde regularităţile lexiconului sunt mult mai pronunţate.

Exemplul 7.1 *În limba engleză, majoritatea verbelor formează persoana a treia singular la indicativ prezent, prin adăugarea sufixului -s la rădăcină (wants, likes, shows,...). Scrierea unei reguli generale care realizează acest lucru se poate face astfel:*

$$(V\ ROOT\ ?r\ SUBCAT\ ?s\ VFORM\ prez\ AGR\ 3s) \longrightarrow$$
$$(V\ ROOT\ ?r\ SUBCAT\ ?s\ VFORM\ ind)(+S)$$

unde $+S$ este o nouă categorie lexicală formată numai din sufixul $-s$.

Dacă ne referim la verbul "to want", această regulă, completată cu linia din lexicon

$$want\ :(V\ \ ROOT\ want$$
$$SUBCAT\ \{_np_vp:inf\ _np_vp:ger\}$$
$$VFORM\ ind)$$

va produce următorul constituient pentru secvenţa de intrare "it want-s":

want : (*V ROOT want*
SUBCAT {*_np_vp : inf _np_vp : ger*}
VFORM pres)
AGR 3s)

Exemplul 7.2 *O altă regulă va genera constituienţii pentru indicativ trecut persoanele I şi II care, pentru limba engleză, au aproape totdeauna aceeaşi formă cu participiul.*

(*V ROOT ?r SUBCAT ?s VFORM prez AGR* (1*s* 2*s* 1*p* 2*p* 3*p*)) ⟶
(*V ROOT ?r SUBCAT ?s VFORM ind*)

Această regulă funcţionează numai pentru verbele regulate. Aplicată fără discernământ, ea poate conduce la forme greşite. De exemplu, verbul "to be" nu poate folosi această regulă; propoziţia

We be at the store.

este greşită. Nu mai vorbim de limba română, unde numărul verbelor neregulate este semnificativ mai mare.

Pentru aceste situaţii - cum este cea din Exemplul 7.2 - se introduce o caracteristică suplimentară care va identifica verbele neregulate. Anume, vom avea o caracteristică binară *IRREG-PRES* pentru verbele care au forme neregulate la indicativ prezent. În acest fel, regula de mai sus se scrie corect:

(*V ROOT ?r SUBCAT ?s VFORM prez AGR* (1*s* 2*s* 1*p* 2*p* 3*p*)) ⟶
(*V ROOT ?r SUBCAT ?s VFORM ind IRREG – PRES–*)

Folosind valoarea de eroare pentru caracteristici, această caracteristică va fi specificată numai în cazul verbelor neregulate. Similar se procedează în cazul realizării timpului trecut sub formă neregulată (pentru limba română - perfectul simplu): caracteristica binară va fi *IRREG-PAST*; sau a participiului: *EN-PASTPRT* pentru verbele care au terminaţia la participiu - *en* în loc de - *ed* (similar limbii române pentru unele verbe: *a fi - fost, a scoate - scos*). Toate aceste caracteristici restricţionează aplicarea regulilor lexicale standard şi toate formele neregulate sunt adăugate explicit în lexicon.

In Tabelul 7.5 este definită o gramatică pentru câteva astfel de reguli, aplicate în declinarea substantivelor şi conjugarea verbelor din limba engleză (am folosit însă termenii din gramatica limbii române - *participiu, gerunziu*).

Fiind dat un set mare de astfel de caracteristici, sarcina de a scrie elementele unui lexicon devine foarte dificilă. De aceea este necesară utilizarea de diverse mecanisme care să ajute la simplificarea definiţiilor.

O primă tehnică - valoarea de eroare pentru caracteristici (care poate fi eventual omisă) - a fost deja menţionată.

O altă tehnică constă în definirea de pachete de caracteristici, notate cu câte un simbol.

În Tabelul 7.6 este construit un mic lexicon. El conţine trei intrări pentru cuvântul *saw*: ca substantiv (*fierăstrău*), verb regulat (*a tăia*) şi trecutul - forma neregulată a verbului *to see (a vedea)*.

Cu acest lexicon şi gramatica din Tabelul 7.5 se pot genera şi alte cuvinte, cum ar fi: *been*, *being*, *cries*, *cried*, *dogs*, *saws* (două interpretări), *sawed*, *sawing*, *seen*, *seeing*, *seeds*, *wants*, *wanting*, *wanted*.

Adesea un cuvânt are mai multe interpretări care folosesc diverse linii din lexicon şi diferite reguli lexicale ale gramaticii. Cuvântul *saws* de exemplu, transformat în secvenţa *saw+s*, poate fi pluralul unui substantiv (via producţia 7 şi prima definiţie pentru *saw*), sau indicativ prezent, persoana a treia singular a verbului *saw* (via producţia 1 şi a doua definiţie pentru *saw*). De remarcat că producţia 1 nu se poate aplica celei de-a treia definiţii, deoarece la aceasta *VFORM* nu este *ind*.

Prelegerea 8

Utilizarea caracteristicilor în analiză

8.1 Generarea gramaticilor cu caracteristici

Vom construi aici o gramatică simplă bazată pe sistemele de caracteristici şi lexiconul (de limbă engleză) dezvoltat în cursul trecut. Ea va accepta propoziţii cum ar fi:

The man cries.	*The men cry.*
The man saw the dogs.	*He wants the dog.*
He wants to be happy.	*He wants the man to see the dog.*
He is happy to be a dog.	

şi va respinge propoziţii de forma:

(*) *The men cries.*	(*) *The man cry.*
(*) *The man saw to be happy.*	(*) *He wants.*
(*) *He wants the man saw the dog.*	

Înainte de a dezvolta această gramatică, vom face câteva convenţii suplimentare.

Deoarece este foarte dificil să scriem regulile gramaticale însoţite de toate caracteristicile necesare, vom căuta să exploatăm anumite regularităţi în folosirea lor. Astfel, multe valori de caracteristici sunt specifice unei singuri caracteristici (de exemplu, valoarea *inf* poate apare numai în caracteristica *VFORM*, iar *_np_vp:inf* aparţine numai caracteristicii *SUBCAT*). De aceea, nu va fi nici o ambiguitate dacă eliminăm numele caracteristicii în situaţiile când nu există ambiguităţi.

Aceste valori unice vor fi listate folosind paranteze pătrate. Deci, de exemplu

$$(VP\ SUBCAT\ inf)$$

va fi abreviată în

$$VP[inf].$$

De asemenea, pentru caracteristicile binare putem introduce o simplificare de notaţie: dacă B este o caracteristică binară, atunci constituientul $(C\ B\ +)$ va fi notat $C[+B]$.

Anumite caracteristici trebuie să verifice restricţia ca valoarea definită la început să coincidă cu valoarea unui subconstituient; ele sunt numite *caracteristici directoare*. De exemplu, în toate regulile *VP*, valorile caracteristicilor *VFORM* şi *AGR* au această proprietate, ca în regula:

(*VP VFORM ?v AGR ?a*) ⟶
(*V VFORM ?v AGR ?a SUBCAT* _np_vp : *inf*)
(*NP*)
(*VP VFORM inf*)

Dacă am putea scrie caracteristicile directoare separat de reguli, la cerere, sistemul le poate adăuga automat. Astfel, dacă *VFORM* şi *AGR* sunt declarate caracteristici directoare, regula de sus poate fi abreviată în:

VP ⟶ (**V SUBCAT _np_vp : inf**) *NP* (*VP VFORM inf*)

Constituientul director al unei reguli a fost indicat prin o scriere îngroşată.

Combinând toate aceste prescurtări convenite, regula anterioară va fi scrisă:

VP ⟶ *V*[**_np_vp : inf**] *NP VP*[*inf*]

In Tabelul 8.1 este definită o gramatică simplă. Exceptând producţiile 1 şi 2 care trebuie să asigure acordul de număr, toate celelalte restricţii de caracteristici pot folosi abrevierile definite anterior.

Tabelul 8.1:

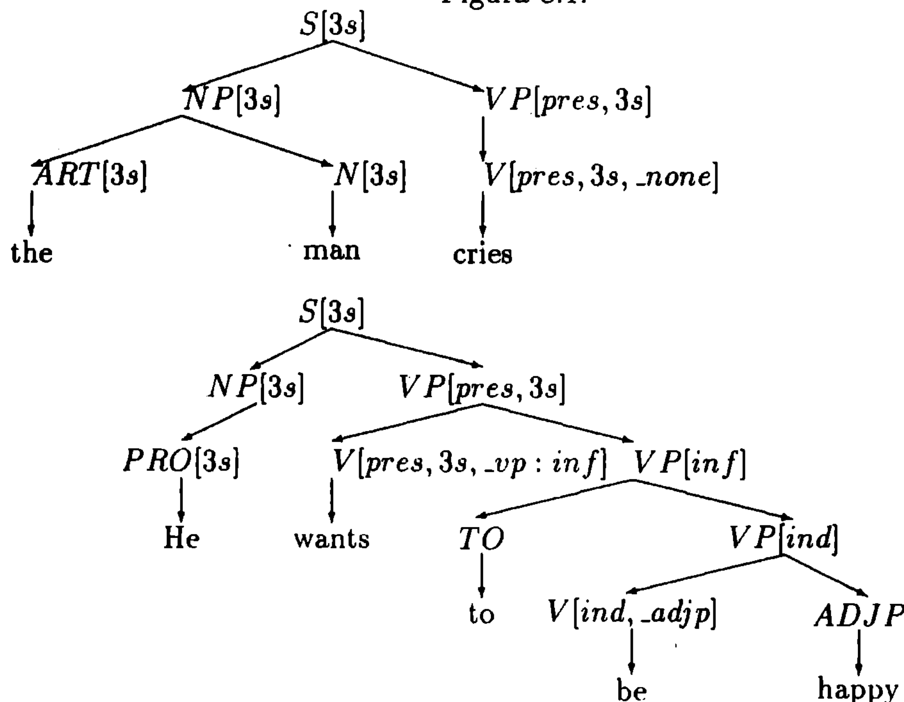
1.	*S*[−*inv*]	⟶	(*NP AGR ?a*) (**VP** [{**prez past**}] **AGR ?a**)
2.	*NP*	⟶	(*ART AGR ?a*) (**N AGR ?a**)
3.	*NP*	⟶	**PRO**
4.	*VP*	⟶	*V*[**_none**]
5.	*VP*	⟶	*V*[**_np**] *NP*
6.	*VP*	⟶	*V*[**_vp : inf**] *VP*[*inf*]
7.	*VP*	⟶	*V*[**_np_vp : inf**] *NP VP*[*inf*]
8.	*VP*	⟶	*V*[**_adjp**] *ADJP*
9.	*VP*[*inf*]	⟶	**TO** *VP*[*ind*]
10.	*ADJP*	⟶	**ADJ**
11.	*ADJP*	⟶	**ADJ**[**_vp : inf**] *VP*[*inf*]
			Caracteristici directoare pentru *S, VP: VFORM, AGR*
			Caracteristici directoare pentru *NP: AGR*

Această gramatică este o scriere prescurtată a gramaticii următoare:

1. (*S INV* – *VFORM ?v{pres past} AGR ?a*) ⟶
   (*NP AGR ?a*) (**VP VFORM ?v{pres past} AGR ?a**)
2. (*NP AGR ?a*) ⟶ (*ART AGR ?a*) (**N AGR ?a**)
3. (*NP AGR ?a*) ⟶ (**PRO AGR ?a**)
4. (*VP AGR ?a VFORM ?v*) ⟶ (**V SUBCAT _none AGR ?a VFORM ?v**)
5. (*VP AGR ?a VFORM ?v*) ⟶ (**V SUBCAT _np AGR ?a VFORM ?v**) *NP*
6. (*VP AGR ?a VFORM ?v*) ⟶
   (**V SUBCAT _vp : inf AGR ?a**) (*VP VFORM inf*)
7. (*VP AGR ?a VFORM ?v*) ⟶
   (**V SUBCAT _np_vp : inf AGR ?a VFORM ?v**) *NP* (*VP VFORM inf*)
8. (*VP AGR ?a VFORM ?v*) ⟶
   (**V SUBCAT _adjp AGR ?a VFORM ?v**) *ADJP*
9. (*VP SUBCAT inf AGR ?a VFORM inf*) ⟶
   (**TO AGR ?a VFORM inf**) (*VP VFORM ind*)
10. *ADJP* ⟶ **ADJ**
11. *ADJP* ⟶ **ADJ (SUBCAT _vp : inf)** (*VP VFORM inf*)

Forma prescurtată este de asemenea foarte utilă în reprezentarea arborilor de analiză. De exemplu, în Figura 8.1 sunt reprezentaţi arborii de analiză pentru două propoziţii date anterior ca exemplu; se arată astfel că ele sunt construcţii corecte din punct de vedere al gramaticii definite în Tabelul 8.1.

Figura 8.1:



Să vedem motivele pentru care propoziţiile date ca exemple negative la începutul prelegerii, nu sunt acceptate de această gramatică.

The men cries nu verifică acordul de număr al regulii 1. Constituientul *NP* pentru *the men* are valoarea *AGR 3p* în timp ce *VP cries* are valoarea *AGR 3s*; deci

regula 1 nu se poate aplica.

La fel, *The man cry* nu este acceptat deoarece *the man* are valoarea *AGR 3s* iar *VP cry* are ca valoare *AGR* o variabilă cu valori în domeniul {*1s 2s 1p 2p 3p*}.

Propoziţia *The man saw to be happy* nu este acceptată deoarece verbul *saw* are valoarea *SUBCAT _np*; deci numai regula 5 poate fi folosită pentru a construi un *VP*. Dar regula 5 cere un complement *NP*, imposibil pentru secvenţa *to be happy*.

Propoziţia *He wants* nu este acceptată deoarece verbul *wants* are valoarea *SUBCAT* din domeniul {*_np _vp:inf _np_vp:inf*}, deci pot fi folosite numai regulile 5, 6 sau 7 pentru a construi un *VP*. Toate aceste reguli cer însă un complement nevid de un tip oarecare.

Propoziţia *He wants the man saw the dog* nu este acceptată din motive similare: din nou, *wants* face posibilă aplicarea regulilor 5, 6 sau 7. 5 şi 6 nu se pot aplica, iar 7 cere un *NP* şi un *VP[inf]*. Construcţia *the man* dă *NP*-ul cerut, dar *saw the dog* nu este un *VP[inf]*; în particular *saw the man* este un *VP* corect, dar restricţia sa *VFORM* va fi *past*, nu *inf*.

8.2 Analiza sintactică folosind caracteristici

Algoritmii de analiză dezvoltaţi anterior pentru gramatici independente de context pot fi extinşi la aceste clase noi de gramatici, prin generalizarea regulilor la constituienţi.

Reamintim, algoritmul bottom-up dădea posibilitatea extensiei arcelor active cu constituienţi noi. Astfel, un constituient X putea extinde un arc de forma $C \longrightarrow C_1 \dots C_i \bullet X \dots C_n$, producând un arc nou de forma $C \longrightarrow C_1 \dots C_i X \bullet \dots C_n$.

O operaţie similară poate fi făcută în cazul gramaticilor cu caracteristici; aici însă analizorul trebuie ca - înainte de a face extensia cu X - să instanţieze variabilele din arcul originar.

Exemplul 8.1 *Să considerăm regula:*

1. (*NP AGR ?a*) ⟶ •(*ART AGR ?a*) (*N AGR ?a*)

Ea spune că se poate defini un NP dintr-un ART şi un N, dacă toate cele trei componente sunt în "rezonanţă" pe caracteristica AGR. Nu există nici o restricţie pentru alte caracteristici pe care le pot avea în plus NP,ART şi N. Deci, dacă considerăm constituienţii pentru această regulă, doar caracteristica AGR este cea care contează; celelalte pot fi ignorate.

Fie de exemplu construcţia "un pahar". Pentru început, va trebui să considerăm extensia arcului 1 *cu constituientul*

2. (*ART ROOT UN AGR* 3*s*)

Pentru a face arcul 1 *aplicabil, variabila ?a trebuie instanţiată cu 3s, ceea ce duce la:*

3. (*NP AGR* 3*s*) ⟶ •(*ART AGR* 3*s*) (*N AGR* 3*s*)

Deoarece caracteristica din regulă este şi în constituientul 2, *arcul poate fi extins:*

4. (*NP AGR* 3*s*) ⟶ (*ART AGR* 3*s*) • (*N AGR* 3*s*)

Să încercăm acum extensia acestui arc cu constituientul corespunzător cuvântului "pahar":

5. (*N ROOT PAHAR*1 *AGR* 3*s*)

Extensia este posibilă pentru că valoarea caracteristicii AGR este aceeaşi. Deci:
6. (*NP AGR* 3*s*) ⟶ (*ART AGR* 3*s*) (*N AGR* 3*s*)•
Aceasta înseamnă că în final analizorul a găsit un constituient de forma (*NP AGR* 3*s*).

Algoritmul este următorul:

> Fiind dat un arc *A* în care constituientul de după • este *NEXT*, şi un nou constituient *X* utilizat pentru extensia arcului:
>
> 1. Se caută o instanţiere a variabilelor astfel încât toate caracteristicile specificate în *NEXT* se află în *X*;
> 2. Se crează un nou arc *A*' care este o copie a lui *A*, cu valorile variabilelor determinate la pasul anterior;
> 3. Se actualizează *A*' conform algoritmului de analiză sintactică similar.

Exemplul 8.2 *Revenind la exemplul anterior, fie A arcul* 1 *şi X constituientul ART dat de* 2. *Atunci NEXT va fi* (*ART AGR* ?*a*).
În pasul (1) *NEXT se identifică cu X şi deci* ?*a va fi înlocuit cu* 3*s*.
La pasul (2) *se face o copie a lui A, care este arcul* 3.
În pasul (3) *acest arc este actualizat conducând la arcul* 4.

Când sunt folosite mulţimi de valori pentru variabile, cum ar fi ?*a*{*3s 3p*}, procesul de identificare decurge similar, cu observaţia că valoarea dată variabilei va fi aleasă dintre cele listate.

Dacă atât regula cât şi constituientul conţin variabile, rezultatul este o variabilă având ca domeniu de valori intersecţia mulţimilor de valori asociate regulii, respectiv constituientului.

Exemplul 8.3 *Să considerăm extensia arcului* 1 *cu constituientul*

(*ART ROOT un AGR* ?*v*{3*s* 3*p*})

(corespunzător cuvântului "un"). Pentru aplicare, variabila ?*a va trebui instanţiată cu* ?*v*{3*s* 3*p*}, *conducând la regula:*

(*NP AGR* ?*v*{3*s* 3*p*}) ⟶ (*ART AGR* ?*v*{3*s* 3*p*}) • (*N AGR* ?*v*{3*s* 3*p*})

Acest arc poate fi extins cu (*N ROOT pahar AGR* 3*s*) *deoarece* ?*v*{3*s* 3*p*} *poate primi valoarea* 3*s*; *rezultatul obţinut este identic cu arcul* 6.

De asemenea, aici caracteristicile subconstituienţilor sunt inserate automat de analizor la fiecare extensie a arcului. Valorile folosite pentru subconstituienţi sunt deja conţinute în hartă.

Exemplul 8.4 *Să presupunem că harta conține deja doi constituienți ART1 și N1 pentru cuvintele "un" respectiv "pahar"; constituientul adăugat în hartă pentru secvența "un pahar" este:*

(*NP* *AGR* 3*s*
1 *ART*1
2 *N*1)

unde

*ART*1 = (*ART ROOT un AGR* {3*s* 3*p*}), *N*1 = (*N ROOT pahar AGR* {3*s*}).

Caracteristica AGR a lui ART1 nu a fost schimbată; deci ea poate fi folosită și de alte interpretări pentru care este posibilă valoarea 3p. Orice algoritm de analiză bottom-up construit anterior poate fi folosit lucrând cu o gramatică augmentată în acest mod.

Figura 8.2:

*S*1	*CAT S* *AGR* 3*s* *VFORM prez* *INV* – 1 *NP*1 2 *VP*3						
		*VP*3	*CAT VP* *VFORM prez* *AGR* 3*s* 1 *V*1 2 *VP*2				
				*VP*2	*CAT VP* *VFORM inf* 1 *TO*1 2 *VP*1		
*NP*1	*CAT NP* *AGR* 3*s* 1 *PRO*1					*VP*1	*CAT VP* *VFORM ind* 1 *V*2
PRO	*CAT PRO* *AGR* 3*s*	*V*1	*CAT V* *ROOT want* *VFORM prez* *AGR* 3*s* *SUBCAT* {_*np*, *vp* : *inf*, _*np*_*vp* : *inf*}	*TO*1	*CAT TO*	*V*2	*CAT V* *ROOT cry* *VFORM ind* *SUBCAT* _*none*
He		*wants*		*to*		*cry*	

Exemplul 8.5 *Figura 8.2 conține o hartă finală obținută din analiza propoziției*

He wants to cry.

folosind gramatica din Tabelul 8.1 (pentru care s-a construit şi derivarea din Figura 8.1).

În această construcţie, constituientul NP1 a fost generat cu regula 3

3 (*NP AGR ?a*) ⟶ (*PRO AGR ?a*)

Pentru a se identifica cu constituientul PRO1, variabila ?a trebuie instanţiată cu 3s; aceasta conduce la constituientul

*NP*1 : (*CAT NP*
AGR 3*s*
1 *PRO*1)

Constituientul VP1 se obţine folosind regula 4, anume:

4 (*VP AGR ?a VFORM ?v*) ⟶ (*V SUBCAT _none AGR ?a VFORM ?v*)

Pentru a corespunde cu V2, în membrul drept variabila ?v trebuie instanţiată cu "ind"; caracteristica AGR a lui V2 nu este definită, aşa că se pune −. Noul constituient este

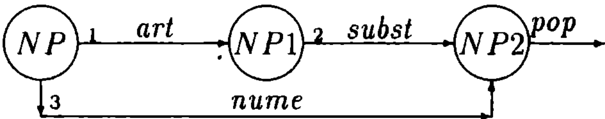
*VP*1 : (*CAT VP*
AGR −
VFORM ind
1 *V*2)

8.3 Reţele de tranziţie augmentate

Caracteristicile pot fi adăugate şi reţelelor de tranziţie recursive, conducând la *reţele de tranziţie augmentate* (**ATN**); în acest caz, caracteristicile poartă numele de *regiştri.* Structurile de constituienţi sunt create permiţând fiecărei reţele să aibă un set de regiştri. De fiecare dată când este activată o reţea, se crează un set nou de regiştri. Odată cu traversarea reţelei, aceşti regiştri capătă valori prin acţiuni asociate fiecărui arc. La părăsirea reţelei, regiştrii sunt aranjaţi astfel încât să formeze un constituient având trecut drept *CAT* numele reţelei.

Exemplul 8.6 *Să luăm reţeaua NP simplă din Figura 8.3:*

Figura 8.3:



Arc	Test	Acţiuni
1	*none*	$DET := *$ $AGR := AGR_*$
2	$AGR \cup AGR_*$	$HEAD := *$ $AGR := AGR \cup AGR_*$
3	*none*	$NUME := *$ $AGR := AGR_*$

Orice **ATN** *foloseşte un mecanism special pentru a extrage rezultatul în urma traversării unui arc. Când se parcurge un arc - de exemplu arcul* 1 *- în prima fază constituientul definit în cuvântul de intrare este salvat într-o variabilă specială numită* $*$. *Acţiunea* $DET := *$ *asignează apoi acest constituient registrului* DET. *A doua acţiune a arcului,* $AGR := AGR_*$ *asignează registrului* AGR *al reţelei valoarea registrului* AGR *din constituientul aflat în* $*$.

Verificarea corespondenţelor corecte este realizată prin teste.

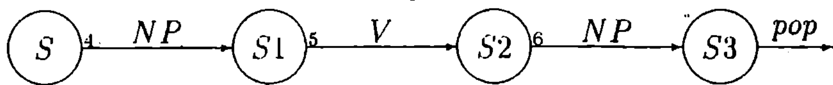
Definiţia 8.1 *Un test este un predicat asociat unei expresii, care ia valoarea "adevăr" dacă rezultatul evaluării expresiei este o valoare nevidă, şi "fals" dacă rezultatul este mulţimea vidă sau nil.*

Dacă un test este incorect, arcul respectiv nu este traversat. Testul pentru arcul 2 din exemplu arată că arcul respectiv poate fi traversat numai dacă caracteristica AGR a reţelei are intersecţia nenulă cu registrul AGR al noului cuvânt (constituientul *subst* din $*$).

În cazul arcelor *push*, caracteristicile sunt tratate similar.

Exemplul 8.7 *Constituientul construit prin traversarea reţelei NP este returnat ca valoare* $*$; *Deci, în gramatica din Figura 8.4*

Figura 8.4:



Arc	Test	Acţiuni
4	*none*	$SUBJ := *$
5	$AGR_{SUBJ} \cap AGR_*$	$MAIN - V := *$
		$AGR := AGR_{SUBJ} \cap AGR_*$
6	*none*	$OBJ := *$
		$AGR := AGR_*$

acţiunea arcului 4, $SUBJ := *$ *va asigna registrului* $SUBJ$ *constituientul întors de reţeaua NP. Testul arcului* 5 *este corect numai dacă registrul* AGR *al constituientului din registrul* $SUBJ$ *are intersecţie nevidă cu registrul* AGR *al noului constituient (verb). Acest test asigură concordanţa subiect-predicat.*

Pe baza lexiconului din prelegerea anterioară, acest ATN acceptă propoziţiile:

The dog cried.
The dogs saw Jack.
Jack saw the dogs.

Exemplul 8.8 *Să detaliem una din aceste propoziţii, de exemplu "The dog saw Jack". Analiza ei pe baza celor două reţele definite mai sus este:*

Intrare: $_1$ *The* $_2$ *dog* $_3$ *saw* $_4$ *Jack* 5

A. Reţeaua S:

Pas	Nod	Poziţie	Arc traversat	Set de regiştri
1	*S*	1	*arc* 4 *corect* *(pentru apel recursiv* *vezi analiza de jos)*	*SUBJ* ← (*NP DET the* *HEAD dog* *AGR* 3*s*)
5	*S*1	3	*arc* 5 *(verifică* 3*s* ∩ 3*s*)	*MAIN* − *V* ← *saw* *AGR* ← 3*s*
6	*S*2	4	*arc* 6 *(pentru apel recursiv* *vezi analiza de jos)*	*OBJ* ← (*NP NUME Jack* *AGR* 3*s*)
9	*S*3	5	*arc 'pop' corect*	*scoate:* ((*S SUBJ* (*NP DET the* *HEAD dog* *AGR* 3*s*) *MAIN* − *V saw* *AGR* 3*s* *OBJ* (*NP NUME Jack* *AGR* 3*s*))

B. Primul apel al reţelei NP: *(arc 4)*

Pas	Nod	Poziţie	Arc traversat	Set de regiştri
2	*NP*	1	1	*DET* ← *the* *AGR* ← {3*s* 3*p*}
3	*NP*1	2	2*(verifică* {3*s* 3*p*} ∩ 3*s*)	*HEAD* ← *dog*
4	*NP*2	*pop*		*întoarce:* (*NP DET the* *HEAD dog* *AGR* 3*s*)

C. Al doilea apel al reţelei NP: *(arc 6)*

Pas	Nod	Poziţie	Arc traversat	Set de regiştri
7	*NP*	4	3	*NUME* ← *Jack* *AGR* ← 3*s*
8	*NP*2	*pop*	*întoarce:*	(*NP NUME Jack* *AGR* 3*s*)

8.4 Gramatici de unificare

După cum s-a văzut, caracteristicile sunt deosebit de utile în generalizarea noţiunilor de gramatică independentă de context şi reţea de tranziţie. Rolul lor poate fi extins astfel încât să elimine complet folosirea gramaticilor. În acest fel, o gramatică poate

fi utilizată doar ca un set de restricţii între diverse structuri de caracteristici. Astfel de sisteme sunt cunoscute sub numele de *gramatici de unificare.*

Conceptul de bază într-o gramatică de unificare este acela de *extensie* între două structuri de caracteristici.

Definiţia 8.2 *O structură de caracteristici F_1 extinde (sau "este mai precisă" decât) F_2 dacă orice valoare de caracteristică din F_1 este specificată în F_2.*

Exemplul 8.9 *Structura (CAT V ROOT doarme) extinde structura (CAT V). Pe de-altă parte nici una din structurile*

(CAT V ROOT doarme) şi (CAT V VFORM prez)

nu este extensia celeilalte.

Definiţia 8.3 *Două structuri* **unifică** *dacă există o structură de caracteristici care este o extensie a ambelor.*

Structura de caracteristici minimală cu această proprietate se numeşte **cel mai general unificator**.

Exemplul 8.10 *Cel mai general unificator al celor două structuri din exemplul anterior este*

(CAT V ROOT doarme VFORM prez)

În schimb, structurile (CAT V AGR 3s) şi (CAT V AGR 3p) nu se pot unifica.

Acest neajuns se elimină dacă putem aloca şi seturi de valori de caracteristici (de exemplu {3s 3p}). Atunci putem spune că, de exemplu (AGR 3s) extinde (AGR {3s 3p}).

Folosind aceste idei, se poate defini o gramatică numai în termeni de unificare. O regulă cum ar fi

$$S \longrightarrow NP\ VP$$

(uzuală în gramaticile pentru limbaje naturale) poate fi exprimată într-o gramatică de unificare astfel:

$$X_0 \longrightarrow X_1 X_2 \qquad \begin{array}{l} CAT_0 = S \\ CAT_1 = NP \\ CAT_2 = VP \\ AGR_0 = AGR_1 = AGR_2 \\ VFORM_0 = VFORM_2 \end{array}$$

Aceasta spune că un constituient X_0 poate fi construit dintr-o sceventă de doi constituineţi X_1 şi X_2, dacă *CAT* pentru X_0 este *S*, pentru X_1 este *NP*, iar pentru X_2 este *VP*; în plus, trebuie ca valorile *AGR* ale celor trei constituienţi să fie identice, la fel ca şi valorile *VFORM* pentru X_0 şi X_2.

Dacă valoarea *CAT* este specificată totdeauna, regula poate fi scrisă mai concis (folosind valorile *CAT* în scrierea producţiei, iar indicele 0 este omis):

$$S \longrightarrow NP\ VP \qquad \begin{array}{l} AGR = AGR_1 = AGR_2 \\ VFORM = VFORM_2 \end{array}$$

În acest fel este posibilă rescrierea oricărei gramatici augmentate.

Exemplul 8.11 *Să exemplificăm afirmaţia de mai sus cu câteva reguli din gramatica definită în Tabelul 8.1:*

1'.	$S \longrightarrow NP\ VP$	$AGR = AGR_1 = AGR_2$ $VFORM = VFORM_2$
2'.	$NP \longrightarrow ART\ N$	$AGR = AGR_1 = AGR_2$
8'.	$VP \longrightarrow V\ ADJP$	$SUBCAT_1 = _adjp$ $VFORM = VFORM_1$ $AGR = AGR_1$
10'.	$ADJP \longrightarrow ADJ$	

Reprezentarea structurilor de caracteristici sub formă de *DAG*

Formalismul de unificare poate fi definit mai precis reprezentând structurile de caracteristici sub formă de grafuri orientate aciclice - pe scurt *dag* (Directed Acyclic Graphs). Fiecare constituient şi valoare formează un nod, iar caracteristicile sunt reprezentate prin arce marcate.

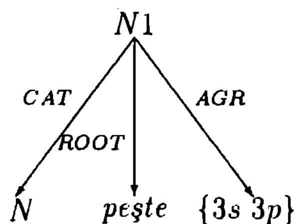
Atenţie ! A nu se face confuzie între *dag* şi *tag* - Tree Adjoining Grammars, introduşi anterior pentru modelări lexicale şi sintactice.

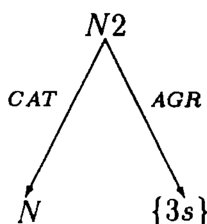
Exemplul 8.12 *Pentru constituienţii*

$N1$: (CAT N
$ROOT$ *peşte*
AGR {3s 3p})

$N2$: (CAT N
AGR 3s)

se pot construi dag-urile:





Trecerea de la un mod de realizare la altul este evidentă.

Unificarea a două structuri de caracteristici este definită sub forma unui algoritm de unificare a două grafuri. Anume:

Algoritmul 8.1

Intrare: *Două dag-uri având ca rădăcini nodurile N_i respectiv N_j.*

Ieşire: *Un dag corespunzător unificării celor două dag-uri de la intrare.*

Algoritm:

1. *Dacă N_i coincide cu N_j, atunci întoarce N_i; Stop.*

2. *Dacă N_i şi N_j sunt noduri terminale, se compară marcajele lor; dacă intersecţia acestora este nevidă, se construieşte un nou nod care are ca marcaj această intersecţie. Altfel, dag-urile nu se pot unifica.*

3. *Dacă N_i şi N_j nu sunt noduri terminale, atunci se crează un nod nou N. Pentru fiecare arc marcat cu F de la nodul N_i la nodul NF_i:*

   (a) *Dacă există un arc marcat cu F de la nodul N_j la NF_j, atunci se unifică recursiv nodurile NF_i cu NF_j. Se construieşte un arc marcat cu F de la N la rezultatul acestei construcţii recursive.*

   (b) *Dacă din N_j nu există nici un arc marcat cu F, se construieşte un arc marcat cu F de la N la NF_i.*

   (c) *Pentru fiecare arc marcat cu F de la N_j la NF_j ce nu are arc similar care pleacă din N_i, se crează un arc nou marcat cu F de la N la NF_j.*

Exemplul 8.13 *Aplicând acest algoritm nodurilor $N1$ şi $N2$ date anterior, se obţine constituientul*

$$N3:\ (CAT\ N\ ROOT\ \text{peşte}\ AGR\ 3s)$$

Se poate construi acum un algoritm de generare a constituienţilor folosind ecuaţiile de unificare bazate pe dag-uri.

Algoritmul 8.2
Intrare:

- *Producţia $X_0 \longrightarrow X_1 \ldots X_n$;*
- *$SC_1, SC_2, \ldots, SC_n$ - subconstituienţii corespunzători lui $X_1, X_2, \ldots, X_n$;*
- *Un set de ecuaţii caracteristice de forma $F_i = V$, unde F_i este caracteristica F a subconstituientului i.*

Ieşire: *Un dag care satisface toate ecuaţiile caracteristice.*
Algoritm:

1. *Se generează un nod CC_0 ca rădăcină a noii structuri;*
2. *Pentru fiecare $i, 1 \leq i \leq n$:*
   - *Se face o copie a dag-ului de rădăcină SC_i şi se notează cu CC_i rădăcina acestei copii;*
   - *Se trasează un arc marcat cu i de la CC_0 la CC_i.*
3. *Pentru fiecare ecuaţie caracteristică de forma $F_i = V$ unde V este o valoare, se urmăreşte legătura F de la nodul CC_i la un nod N_i şi se unifică N_i cu V.*
4. *Pentru fiecare ecuaţie caracteristică (de forma $F_i = G_j$):*

   (a) *Dacă există o legătură F din CC_i şi o legătură G_j din CC_j, atunci:*

      i. *se urmăreşte F până la un nod N_i şi G până la un nod N_j;*

      ii. *se unifică N_i cu N_j folosind algoritmul anterior şi se crează astfel un nod nou X;*

      iii. *toate arcele care intrau în N_i sau N_j sunt aduse în X.*

(b) *Dacă nu există nici o legătură F din CC_i dar există o legătură G de la CC_j la un nod N_j, se crează o legătură F de la CC_i la N_j;*

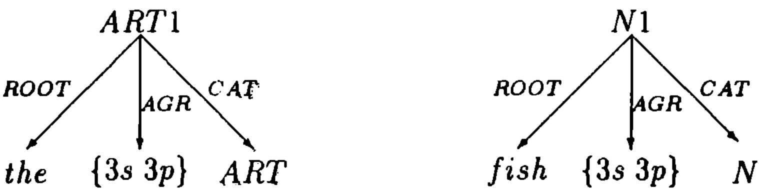
(c) *Dacă nu există nici o legătură G din CC_j dar există o legătură F de la CC_i la un nod N_i, se crează o legătură G de la CC_j la N_i.*

Exemplul 8.14 *Să presupunem că sunt definiţi constituienţii*

ART_1 : (CAT ART
ROOT the
AGR {3s 3p})

N1 : (CAT N
ROOT fish
AGR {3s 3p})

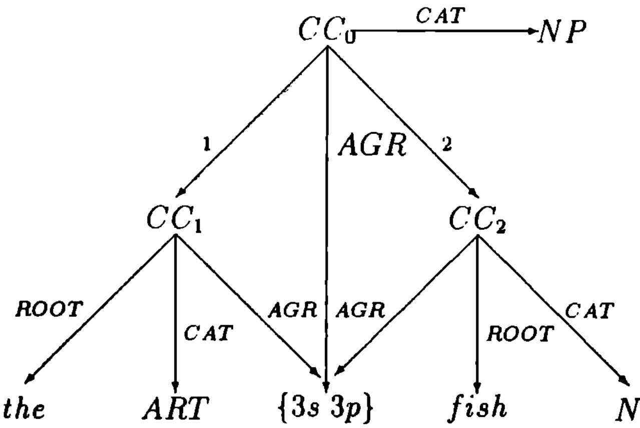
sau, în notaţie de dag-uri:



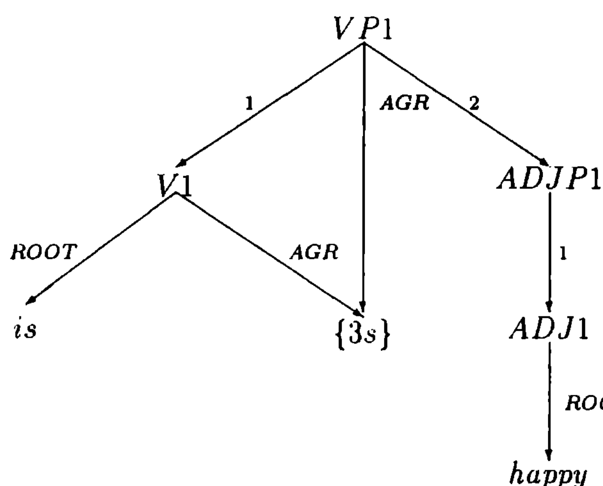
Un NP pentru expresia "the fish" poate fi acum construit folosind regula 2'. Ecuaţiile caracteristice sunt:

$CAT_0 = NP$
$CAT_1 = ART$
$CAT_2 = N$
$AGR_0 = AGR_1 = AGR_2$

Algoritmul va realiza constituientul reprezentat de dag-ul:

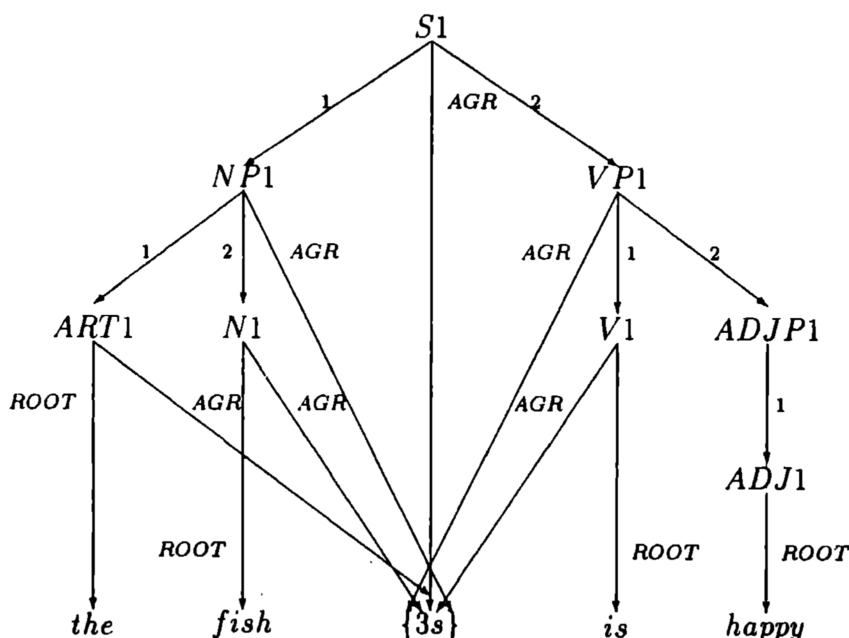


Exemplul 8.15 *Să presupunem că expresia verbală (VP) "is happy" este analizată similar şi reprezentată prin dag-ul (pentru simplificare nu au fost trasate arcele CAT):*



Pe baza acestor analize şi a regulii 1 *din gramatica dată în Tabelul 8.1, se poate construi analiza propoziţiei*

The fish is happy:



Prelegerea 9

Semantici şi logică formală

Definirea precisă a noţiunilor de *semantică* şi *a înţelege* este surprinzător de dificilă, pentru că aceşti termeni sunt folosiţi de obicei cu diverse utilizări, adesea înrudite. Astfel, există întrebuinţări ale verbului *a înţelege* care nu au nici o legătură cu lingvistica. De exemplu:

Deşi nu vorbeşte, Clara înţelege limba spaniolă.

Îi înţeleg durerea.

Ceea ce dorim să folosim este mai apropiat de utilizarea acestui cuvânt în propoziţia

Prin "singur" se înţelege *neînsoţit de nimeni altcineva.*

Aceleaşi probleme apar relativ la termenul *a însemna.*

În general, în semantică se defineşte înţelesul unui cuvânt în termenii altor cuvinte.

Pentru a mări claritatea noţiunilor, vom dezvolta un instrument mai precis, un limbaj formal în care să se poată specifica înţelesul fără să ne referim din nou la limbajul însuşi. Dar chiar şi aşa, definirea noţiunii de *înţeles a unei propoziţii* este dificilă. De exemplu, pe stradă, cineva te poate întreba *"Aveţi un ceas ?"*. Potrivit întrebării, se poate răspunde *Da/Nu* în funcţie de situaţia în care posezi sau nu un astfel de obiect (acasă sau la mână). De fapt, scopul întrebării puse este acela de a afla cât este ceasul. Deci, semnificaţia acestei chestionări poate fi, în funcţie de context, aceea de a afla dacă ai un ceas sau dacă poţi spune cât este ceasul.

O întrebare importantă care se pune este aceea dacă se poate defini înţelesul unei propoziţii independent de contextul în care a fost ea formulată. Adică, există un nivel de reprezentare a cunoştinţelor în care propoziţia *Aveţi un ceas ?* are un singur înţeles dar poate fi folosită în diverse scopuri ? Răspunsul nu este deloc simplu, dar, prin realizarea acestui deziderat vor apare o serie de avantaje în prelucrarea limbajului natural.

Primul argument este *modularitatea.* Dacă o astfel de diviziune poate fi făcută, atunci putem studia înţelesul unei propoziţii în detaliu, fără a ne mai complica cu tratarea modului ei de folosire. Dacă o propoziţie nu are nici un înţeles independent de context, atunci nu vom putea să separăm studiul limbajului de studiul raţionamentului general uman. Ceea ce, în afară de complicaţiile deosebite aduse în

prelucrarea limbajului, nu se întâmplă totdeauna în practică. De aceea vom folosi termenul de *înţeles* pentru acest sens de tratare a propoziţiilor - independent de context; pentru aspectele dependente de context vom folosi termenul de *utilizare*.

Definiţia 9.1 *Reprezentarea înţelesului independent de context este numită* **formă logică**. *Operaţia de asociere unei propoziţii a formei sale logice este numită* **interpretare semantică**, *iar procesul de asociere unei formei logice în limbajul de reprezentare a cunoştinţelor (KR), se numeşte* **interpretare contextuală**.

Pentru moment vom folosi drept limbaj de reprezentare a cunoştinţelor calculul predicatelor de ordinul I (First-Order Predicate Calculus: *FOPC*); definirea sa formală, riguroasă, este dată în *Anexa* 1.

În mod normal, forma logică nu face parte din limbajul de reprezentare a cunoştinţelor. Aceasta conduce la problema de a stabili poziţia sa în procesul de prelucrare a limbajului natural. O idee folosită cu succes se bazează pe noţiunea de **situaţie** - care este definită ca fiind *o mulţime particulară de circumstanţe ale lumii înconjurătoare.*

De exemplu, la intrarea într-o sală de clasă, suntem în situaţia când există studenţi, un profesor, se fac anumite afirmaţii, se pun întrebări etc., sunt implicate şi anumite obiecte, cum ar fi tabla, cretă, scaune, ş.a.m.d.

Definiţia 9.2 *Se numeşte situaţie o pereche ordonată* $\mathcal{S} = (\mathcal{O}, \mathcal{R})$ *unde:*

- $\mathcal{O}$ *este o mulţime finită de elemente numite "obiecte", un obiect fiind un cuplu* (x, y)*; orice două obiecte din* $\mathcal{O}$ *diferă cel puţin prin* y.

- $\mathcal{R}$ *este o mulţime finită de elemente numite "relaţii", o relaţie fiind de forma* $(\alpha, y_1, \ldots, y_n), (n \geq 1)$ *cu proprietatea că pentru orice* $i (1 \leq i \leq n)$ *există un obiect de forma* (x, y_i)*;* α *este o relaţie de aritate* n.

Exemplul 9.1 *O situaţie foarte simplă poate fi următoarea:*

$$\mathcal{S}_1 = (\{(MINGE, B0005), (COPIL, C86), (COPIL, C87)\},$$
$$\{(POSED\breve{A}, C86, B0005)\}).$$

Există deci trei obiecte - o minge şi doi copii - codificate prin $B0005, C86$ *respectiv* $C87$ *şi o relaţie binară - aceea că mingea aparţine copilului* $C86$.

Limbajul crează tipuri speciale de situaţii bazate pe modul de utilizare a informaţiei. Deoarece acest mod de lucru va fi detaliat mai târziu, vom prezenta pentru început câteva elemente intuitive. În orice conversaţie sau text, să presupunem că există o situaţie generată de discurs care înregistrează informaţia pentru a o utiliza în continuare. Orice nouă afirmaţie este interpretată în raport cu această situaţie, producând o situaţie nouă care va include informaţia folosită.

Din acest punct de vedere, putem modela "universul discursului" sub forma unui automat cu o infinitate de stări - fiecare stare fiind o situaţie a discursului, iar forma logică este funcţia de tranziţie a automatului care aplică situaţia discursului în care este făcută prezumţia, într-o nouă situaţie, rezultată din această prezumţie.

Exemplul 9.2 *Să presupunem că situaţia prezentată în Exemplul 9.1 a fost descrisă de o afirmaţie anterioară care conţinea mingea şi cărui băiat îi aparţine ea. Prezumţia*

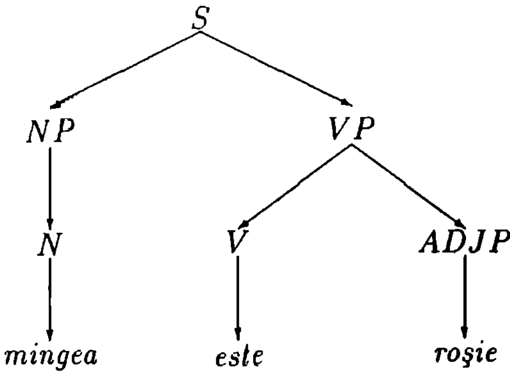
Mingea este roşie.

crează o nouă situaţie, formată din cea anterioară, la care se adaugă faptul că B0005 *are proprietatea ROŞU:*

$$S_2 = (\{(MINGE, B0005), (COPIL, C86), (COPIL, C87)\},$$
$$\{(POSEDĂ, C86, B0005), (ROŞU, B0005)\}.$$

Se poate da şi o interpretare grafică acestei situaţii, în felul următor:

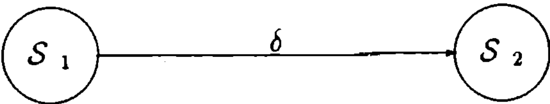
Analiza sintactică:



⇓ *interpretare semantică*

Forma logică: δ = *(ASIGN (ROŞU1* < **b1** *MINGE* >)

Interpretare contextuală:



Situaţia iniţială a discursului *Situaţia actualizată a discursului*

Chiar dacă cea mai mare parte a limbajului este dependentă de context, există totuşi o structură semantică considerabilă a limbajului care se comportă independent de context şi poate fi deci folosită în procesul de interpretare semantică, pentru a produce forma logică. Cea mai mare parte din aceste cunoştinţe semantice constau din tipul de informaţie care poate fi oferit de un dicţionar - proprietăţile semantice de bază ale cuvintelor, ce sensuri diferite sunt posibile pentru fiecare cuvânt, ce sensuri pot fi combinate pentru a forma o structură semantică mai largă, etc.

9.1 Despre FOPC

Se poate considera că logica s-a dezvoltat iniţial ca o notaţie destinată surprinderii proprietăţilor formale ale limbajului şi raţionamentului natural. În consecinţă, multe

proprietăţi structurale au corespondent în limbaj natural. De exemplu, se face deosebire între expresii care identifică obiecte şi expresii care atribuie proprietăţi obiectelor şi identifică relaţiile dintre obiecte. Expresiile substantivale corespund primului tip, iar propoziţiile şi clauzele, celui de-al doilea. În FOPC, aceeaşi distincţie se face între *termeni* (care semnifică obiecte ale universului) şi *propoziţii* (care fac afirmaţii despre obiectele universului).

Există două clase importante de termeni: *constante* şi *funcţii*. Constantele sunt similare practic numelor proprii din limbaj: *Radu*1, *Radu*2, *C*1,... Ele nu semnifică nimic prin simpla enunţare; proprietăţile lor apar numai din formulele care le folosesc.

Apare totuşi o diferenţă majoră între numele proprii din limbajul natural şi constantele din *FOPC*: în timp ce limbajul natural este ambiguu, fiecare simbol din *FOPC* are un singur înţeles. Numele *Ion* poate fi folosit în limbaj natural pentru a face referinţă la mai mulţi oameni diferiţi; acest lucru nu este posibil în logică, unde este necesar câte un nume unic pentru fiecare persoană.

Funcţiile în *FOPC* corespund expresiilor substantivale care se referă la un obiect în termenii anumitor proprietăţi sau relaţii cu alte obiecte, cum ar fi *tatăl lui Ion* sau *prinţul Arabiei*. Ele sunt scrise în *FOPC* ca expresii formate dintr-un nume de funcţie (constantă) urmat de o listă de argumente (termeni), închise între paranteze. Numărul de argumente formează *aritatea* funcţiei; de remarcat că ea nu mai este o constantă (ca în matematică). De exemplu, *tatăl lui Ion* desemnează o funcţie *tată* şi se scrie *tată(Ion1)*. Dar aceeaşi funcţie, în *Tatăl lui Ion şi al Anei* se va scrie prin *tată(Ion1,Ana1)*.

Propoziţiile simple din *FOPC* (în care numele predicatului corespunde unei expresii verbale formată numai din verb) constau dintr-un nume de predicat (constantă) şi o listă de argumente (termeni) închise între paranteze.

Exemplul 9.3 *Pentru propoziţia*

Ion îşi iubeşte tatăl.

se poate construi formula

Iubeşte(Ion1,tata(Ion1)).

Propoziţiile mai complexe sunt construite din cele simple folosind o serie de operatori logici cu corespondent în limbaj natural. Aritatea acestor operatori este fixă, independentă de propoziţiile în care sunt folosite.

Cel mai simplu operator logic este *negaţia*, scrisă $\neg$. Astfel, propoziţiei

Ion nu-şi iubeşte tatăl.

îi poate corespunde formula

$\neg$ *Iubeşte(Ion1,tata(Ion1)).*

Practic, aproape toţi conectorii folosiţi în limbajul natural corespund la diverşi operatori logici din *FOPC*. De exemplu:

α şi β	$\alpha \& \beta$ ("conjuncţie")
α sau β	$\alpha \vee \beta$ ("sau concesiv")
	$\alpha \bigtriangledown \beta$ ("sau exclusiv")
dacă α atunci β	$\alpha \supset \beta$ ("implicaţie")
α dacă şi numai dacă β	$\alpha \equiv \beta$ ("echivalenţă")

În sfârşit, există construcţii în *FOPC* care folosesc *cuantificatori*: $\exists$ (cuantificator *existenţial*) şi $\forall$ (cuantificator *universal*).

Exemplul 9.4 *Următoarele propoziţii exemplifică folosirea celor două tipuri de cuantificatori:*
$\exists x.\ FATĂ(x) \& IUBEŞTE(x, ION1)$
Există o fată care îl iubeşte pe Ion.

$\forall y.FATĂ(y) \supset IUBEŞTE(y, ION1)$:
Toate fetele îl iubesc pe Ion.

Deoarece operatorii logici pot fi legaţi unii de alţii, vor apare ambiguităţi în formule, în funcţie de domeniul de acţiune al fiecărui operator. De exemplu, formula

$$\neg P \& Q$$

poate fi interpretată că fiind conjucţia a două formule ($\neg P$ şi Q) sau ca negaţia formulei $P \& Q$.

În limbajul natural, ei îi corespunde o propoziţie de forma:

Nu merg şi citesc ziarul.

în care nu este clar ce se neagă: doar prima propoziţie simplă - şi atunci semnificaţia este

Stau pe loc şi citesc ziarul,

sau toată propoziţia, ceea ce ar duce la înţelesul:

Stau (şi nu are importanţă ce fac) sau merg - dar atunci nu citesc nici un ziar.

Această ambiguitate se elimină asociind operatorilor logici o listă de priorităţi *de domeniu*; astfel, negaţia are totdeauna cel mai mic domeniu de acţiune posibil, iar disjuncţia - cel mai mare. Atunci, în exemplul de sus $\neg P \& Q \equiv (\neg P) \& Q$.

Aceste priorităţi funcţionează bineînţeles în cazul când nu s-au folosit paranteze pentru delimitarea domeniilor de acţiune ale operatorilor.

Odată cu cuantificatorii s-a introdus şi convenţia de notaţie '.' (punct). El este folosit în formule pentru a indica că domeniul operatorului care îl precede este extins până la sfârşitul formulei. Această convenţie uşurează scrierea, eliminând o serie de paranteze.

Exemplul 9.5 *Propoziţiei:*

Orice pasăre are cuibul ei.

i se poate construi formula logică:
$\forall b.PASĂRE(b) \supset \exists g.CUIB(g) \& APARŢINE(b, g)$

Aceasta este echivalentă cu formula (în care nu s-a folosit convenţia cu punct):

$\forall b(PASĂRE(b) \supset g(CUIB(g) \& APARŢINE(b, g)))$

9.2 Sensurile cuvintelor şi ambiguitatea

Pentru a dezvolta o teorie a semanticii şi a interpretării semantice, va fi nevoie să construim un model structural, aşa cum s-a procedat şi la sintaxă. Acolo s-a definit întâi noţiunea de "*clasă sintactică de bază*", apoi s-au dezvoltat căile prin care aceste clase se pot combina pentru a forma structuri mai largi. În semantică problema este ceva mai complicată, din cauza ambiguităţii. De exemplu, în *Micul dicţionar enciclopedic*, verbul *a da* are 32 definiţii (12 ca verb tranzitiv, 14 ca verb intranzitiv şi 6 ca verb tranzitiv sau intranzitiv, urmat de determinări locale). Uneori pentru el se pot găsi sinonime, cum ar fi: *a oferi, a restitui, a atribui, a renunţa, a se lovi, a ieşi* (fiecare din ele având însă sensuri uşor diferite de verbul *a da*).

Dacă fiecare cuvânt are unul sau mai multe sensuri, atunci va trebui definit un univers al sensurilor deosebit de larg; pentru a-l putea manevra însă, va fi necesară organizarea sa în clase de obiecte cu anumite proprietăţi. Mulţimea acestor clase de obiecte se numeşte *ontologie*. Ideea a fost enunţată de Aristotel, care în scrierile sale definea drept clase majore *substanţa* (obiectele fizice), *cantitatea* (de exemplu *numerele*), *calitatea* (cum ar fi *luminozitatea, culoarea* etc), *relaţia, locul, timpul, poziţia, starea, acţiunea* şi *afecţiunea*. La aceste clase se pot adăuga altele, cum ar fi *evenimentele, ideile, conceptele* şi *planurile*. Dintre acestea se disting câteva clase cu importanţă deosebită: acţiunile, evenimentele şi situaţiile.

Definiţia 9.3 **Evenimentele** *sunt lucruri care apar în universul cunoaşterii pentru a asigura structurarea organizării şi interpretării propoziţiilor.*

Acţiunile *sunt lucruri pe care le fac agenţii (deci cauzează evenimente).*

Situaţiile *se referă la un anumit set de circumstanţe; ele pot fi considerate ca fiind incluse în mulţimea evenimentelor.*

Rezultă că ambiguitatea este o problemă dificilă care trebuie tratată în timpul interpretării semantice.

Definiţia 9.4 *Spunem că un cuvânt este semantic ambiguu dacă îi putem atribui mai multe sensuri.*

Problema se complică dacă vrem să detaliem termenul de *atribuire*. Sunt puţine teste lingvistice care să definească cât mai precis noţiunea de ambiguitate semantică.

Un astfel de test se bazează pe ideea că anumite construcţii sintactice se referă la clase identice de obiecte. De exemplu, propoziţia

Ion are un vas iar Vasile două vase.

are ca semnificaţie faptul că *Ion* şi *Vasile* posedă vase din aceeaşi clasă de obiecte. Nu reiese de exemplu că *Ion* este proprietarul unui vas de pescuit iar cele două vase ale lui *Vasile* sunt vase de bucătărie. Pe de-altă parte, dacă spunem

Mihai are un cal iar Radu are doi cai,

de aici nu reiese că toţi caii din propoziţie au aceleaşi caracteristici (sunt toţi armăsari, au aceeaşi rasă, sunt toţi cai de curse, şi altele). Intuitiv, cuvântul *vas* este ambiguu între sensurile *VAS_NAVĂ* şi *VAS_RECIPIENT*, iar cuvântul *cal* nu este ambiguu între *ARMĂSAR* şi *IAPĂ*, cele două sensuri fiind incluse în semnificaţia

noţiunii *CAL*. Deci, putem deduce de aici că pentru fiecare cuvânt există o ordonare a preciziei sensurilor sale. Această proprietate este denumită **vaguitate**. *CAL* este *vag* deoarece nu distinge între armăsar şi iapă. *ARMĂSAR* este *vag* deoarece nu distinge între mânz şi animal adult. Toate sensurile se pare că posedă un anumit grad de vaguitate şi pot căpăta prin completare un grad mai mare de precizie.

Acelaşi tip de ambiguitate se poate distinge şi în cazul verbelor.

Astfel, propoziţia

Am candidat anul acesta; la fel şi George.

se referă la candidaturile a două persoane pentru atingerea aceluiaşi ţel: alegeri politice, admiterea la facultate, mâna unei fete, sponsorizarea unui proiect; Nu se poate deduce de exemplu că eu am candidat în alegeri iar George la admiterea la facultate. Deci verbul *a candida* este ambiguu între mai multe sensuri: *CANDIDA1* (sens politic), *CANDIDA2* (sens academic), *CANDIDA3* (sens social), şi altele. În schimb verbul *a săruta* este vag deoarece nu specifică locul unde se sărută. Se poate spune

Andrei a sărutat-o pe Andreea; la fel şi Paul.

chiar dacă Andrei a sărutat-o pe frunte iar Paul a sărutat-o pe obraz.

Unele forme de ambiguitate semantică sunt o consecinţă a construcţiei sintactice. Un exemplu uşor de construit se bazează pe domeniul de aplicabilitate al cuantificatorilor. Astfel, în propoziţia

Orice băiat iubeşte un câine.

nu este clar dacă toţi băieţii iubesc acelaşi câine, sau fiecare băiat are preferinţă pentru un anumit câine. Structura sintactică este aceeaşi, dar diferenţa rezultă din domeniile pe care sunt valabili cuantificatorii.

Acestui exemplu i se pot asocia două semnificaţii (în notaţie *FOPC*):

$$\exists d.Câine(d) \& \forall b.Băiat(b) \supset Iubeşte(b,d)$$

$$\forall b.Băiat(b) \supset \exists d.Câine(d) \& Iubeşte(b,d)$$

Un aspect foarte important al semnificaţiei cuvintelor este relaţia dintre sensurile cuvintelor aflate în aceeaşi propoziţie. Adesea sensul corect al unui cuvânt este determinat în mod unic de înţelesul restului propoziţiei.

De exemplu, semnificaţia verbului *a da* este unic determinată în propoziţiile

Petre dă banii pe cărţi.

şi în

Petre dă perdeaua la o parte.

deşi înţelesurile sunt complet diferite. Un rol de bază al interpretării semantice constă tocmai în utilizarea de astfel de restricţii ale propoziţiei, care să conducă la reducerea numărului de sensuri posibile pentru fiecare cuvânt.

9.3 Limbajul formelor logice de bază

Vom defini un limbaj - foarte asemănător cu *FOPC* - care să permită combinarea unităţilor de bază ale *înţelesului* (semnificaţia cuvintelor), pentru a obţine înţelesuri asociate unor expresii mai complexe. Înţelesurile unui cuvânt vor servi ca atomi sau constante ale reprezentării. Aceste constante vor fi clasificate după tipurile de noţiuni pe care le descriu. În particular, constantele care descriu obiecte ale universului (inclusiv cele abstracte, cum sunt *evenimentele* sau *situaţiile*) se numesc *termeni*. Constantele care descriu relaţii şi proprietăţi se numesc *predicate*.

O propoziţie din acest limbaj este formată dintr-un predicat urmat de un număr arbitrar de termeni folosiţi ca argumente.

Exemplul 9.6 *Propoziţia corespunzătoare afirmaţiei:*

Andrei este un băiat,

va fi construită din constantele ANDREI1 (termen) şi BĂIAT1 (predicat); reprezentarea ei este:

(BĂIAT1 ANDREI1).

Prin utilizarea unei alte clase de constante, numite ***operatori (conectori) logici*** se pot realiza propoziţii mai complexe. În general conectorii sunt cei cunoscuţi din logica formală, anume:
- *negaţia* - (NOT)
- *disjuncţia* ($\vee$)
- *conjuncţia* ($\&$)
- *implicaţia* ($\supset$)

precum şi alte forme (*FOPC* are 16 astfel de operatori logici binari).

Similar lor, în limba vorbită există operatorii ***şi, sau, dacă, numai dacă*** etc; deosebirea este aceea că relaţiile induse de limbajul natural prin aceşti conectori sunt mai complexe decât cele din logica formală. De exemplu, conjuncţia ***şi*** corespunde operatorului logic &, dar adesea dă şi o structură temporală evenimentelor, ca în

Am ajuns acasă şi am băut ceva.

în care evenimentul sosirii îl precede pe cel al băuturii. La fel, conectorul ***dar*** operează după aceleaşi reguli logice ca *şi*; aici însă, doilea argument este ceva ce auditoriul nu se aşteaptă să fie adevărat, conform primului argument.

Forma logică generală a propoziţiilor construite folosind conectori este:

$$(< \text{conector} > (< \text{propoziţie} >)(< \text{propoziţie} >)).$$

Exemplul 9.7 *Forma logică pentru aserţiunea*

Paul caută cartea sau Paul caută servieta.

este:

(∨ (CAUTĂ1 PAUL1 CARTEA1)(CAUTĂ1 PAUL1 SERVIETA1))

Cu aceste propoziţii simple - definite anterior - se poate construi înţelesul unui sublimbaj foarte limitat al unei limbi naturale, anume propoziţii formate numai din

verbe simple şi substantive (de obicei nume proprii). Pentru a realiza propoziţii mai complexe, trebuiesc definite şi alte construcţii semantice. O astfel de construcţie, deosebit de importantă, este *cuantificatorul*. În calculul predicatelor de ordinul 1 există doar doi cuantificatori: $\exists, \forall$. Limbajul natural conţine o gamă mult mai vastă de cuantificatori, cum ar fi *toţi, unii, cel mult, multe, câţiva* etc. Pentru a le ataşa cuantificatori, variabilele sunt scrise ca în *FOPC*, cu o singură diferenţă: în logica predicatelor de ordinul 1 se reţine în domeniul de acţiune al cuantificatorului numai semnificaţia variabilei. Deci două apariţii ale aceleiaşi variabile x care apar în două formule diferite (cum ar fi $\exists x.P(x)$ şi $\forall x.Q(x)$) sunt considerate variabile diferite, fără nici o legătură între ele.

În limbajul natural, consideraţiile sunt altele.

Exemplul 9.8 *Să luăm propoziţiile:*

Un om a intrat în cameră.
El s-a suit pe masă.

Prima propoziţie introduce în discuţie un nou obiect, un anumit om. *Suntem tentaţi să formalizăm aceasta folosind cuantificatorul existenţial* $\exists$; *problema este că la acest om - introdus existenţial în prima propoziţie - se face referire prin pronumele* El *în a doua propoziţie. Deci variabila apărută îşi continuă existenţa şi după ce a fost introdusă. De aceea, fiecare introducere a unei variabile în discurs, se realizează asociindu-i un nume unic, nefolosit anterior.*

Cuantificatorii pe limbaj natural au domenii mai restrânse de acţiune şi deci sunt mai complexi.

Astfel, o formulă în *FOPC* de forma $\forall x.P(x)$ este adevărată dacă şi numai dacă $P(x)$ este adevărată pentru orice obiect posibil din domeniu (deci x poate fi orice termen din limbaj).

Astfel de afirmaţii sunt rare în limbajul natural. Aici vom folosi termenul de *cuantificator generalizat*; el este folosit în construcţii de forma:

(< *var_cuantificator* >: (< *restricţie* >)(< *prop_principală* >))

Exemplul 9.9 *Propoziţia "Mulţi câini latră" are forma logică:*
(MULŢI d1*: (CÂINE1* d1*)(LATRĂ1 d1))*·

Aceasta arată că multe obiecte obiecte d1 *care satisfac condiţia (CÂINE1 d1), satisfac şi condiţia (LATRĂ1 d1). De remarcat că această condiţie este complet diferită de formula*

(MULŢI d2*: (LATRĂ1* d2*)(CÂINE1* d2*))*

care defineşte înţelesul propoziţiei "Multe lucruri care latră sunt câini."

O clasă foarte importantă de cuantificatori corespunde articolelor hotărât şi nehotărât. Astfel, afirmaţia

Câinele latră.

are forma logică:

(ART_LE x: *(CÂINE1* x*)(LATRĂ1* x*))*

şi va fi adevărată numai dacă există un câine unic determinat de context, şi acel câine latră. Evident, în orice aplicaţie practică vor fi mai mulţi câini, aşa că folosirea contextului pentru a-l identifica pe cel corect este crucială în înţelegerea propoziţiei.

Folosind diverse restricţii pot rezulta expresii substantiv mai complexe.
De exemplu, propoziţia

Câinele fericit latră.

va folosi o restricţie suplimentară, scrisă sub forma unei conjucţii, anume:
(ART_LE x: *(&(CÂINE1* x*)(FERICIT* x*))(LATRĂ1* x*))*

Ea va fi adevărată numai dacă există un context unic x astfel încât
(&(CÂINE1 x*) (FERICIT* x*))* şi acest x latră.

Pentru a cuantifica formele de plural, trebuiesc introduse alte construcţii. În acest scop, definim un operator unar numit *operator predicat*; el va lua ca argument un predicat şi va produce un nou predicat. Pentru constrcuţia pluralului se foloseşte operatorul predicat *PLUR*.

Exemplul 9.10 *Dacă CÂINE1 este un predicat adevărat pentru orice câine, atunci (PLUR CÂINE1) este un predicat adevărat pentru orice mulţime de câini.*
Deci reprezentarea înţelesului propoziţiei

Câinii latră.

este:
(ART_I x:*((PLUR CÂINE1)* x*)(LATRĂ1* x*))*

Pluralul expresiilor substantivale induce posibilitatea unei noi forme de ambiguitate. Să observăm că înţelesul normal al propoziţiei

Câinii latră.

este acela că există o haită specificată de câini şi fiecare din ei latră. Aceasta este numită *citire distributivă*, pentru că predicatul *LATRĂ1* se distribuie la fiecare element din mulţime.
Să considerăm acum propoziţia

Câinii s-au strâns la colţ.

Aici este un nonsens să afirmăm că fiecare câine s-a strâns; afirmaţia este adevărată pentru întreaga mulţime de câini.
Aceasta se numeşte *citire colectivă*. Unele propoziţii folosesc ambele tipuri de interpretări, deci sunt ambigue.

Exemplul 9.11 *În propoziţia*

Doi oameni şi-au cumpărat maşină.

se poate înţelege că fiecare om şi-a cumpărat maşina lui proprie (citire distributivă) sau că cei doi au cumpărat împreună o singură maşină (citire colectivă).

În sfârşit, ultimele construcţii care pot fi introduse sunt *operatorii modali*, necesari pentru a reprezenta înţelesul verbelor în general, restricţiile de timp în particular. Aceşti operatori sunt destul de asemănători cu operatorii logici, având însă unele deosebiri importante. Anume, termenii din domeniul unui operator modal pot avea o interpretare care diferă de cea normală; aceasta afectează concluziile care se pot trage dintr-o propoziţie.

De exemplu, să presupunem că *Adi* este cunoscut unora sub numele de *Andy*;
există deci două sensuri ale cuvântului, care sunt egale: *ADI1=ANDY2*. În propoziţii construite folosind această supoziţie, nu are importanţă care din cele două constante este folosită: dacă *(FERICIT ANDY2)* este adevărată, atunci *(FERICIT ADI1)* este adevărată şi invers; la fel,
(ORI (VESEL ADI1)(SUPARAT ADI1)),
(ORI (VESEL ANDY2)(SUPARAT ANDY2))
sunt ambele adevărate sau false în acelaşi timp. Aserţiunea nu este însă totdeauna valabilă în interiorul domeniului unui operator modal, cum ar fi *CREDE1*. De exemplu, dacă

Rodica crede că Adi este fericit.

deci
(CREDE RODICA1 (FERICIT ADI1)),
aceasta nu implică neapărat valabilitatea propoziţiei

Rodica crede că Andy este fericit.

adică
(CREDE RODICA1 (FERICIT ANDY2)),

pentru simplul motiv că Rodica poate să nu ştie că *Adi* şi *Andy* sunt una şi aceeaşi persoană.

Deci nu se pot înlocui liber termenii egali, dacă aceştia apar în interiorul domeniului unui operator modal. Această restricţie se numeşte ***eşecul substituţiei în contexte modale.***

O clasă importantă de operatori modali pentru limbajul natural sunt operatorii de timp: *PREZENT, TRECUT, VIITOR*. Deci se pot diferenţia propoziţii ca

Marius vede marea.
Marius a văzut marea.
Marius va vedea marea.

prin:
(PREZENT (VEDE1 MARIUS1 MAREA1)
(TRECUT (VEDE1 MARIUS1 MAREA1)
(VIITOR (VEDE1 MARIUS1 MAREA1)

Aceştia sunt operatori modali deoarece ei verifică eşecul substituţiei.

De exemplu, să considerăm operatorul *TRECUT* şi să luăm două constante *VASILE1* şi *DECAN1*, egale în prezent (ceea ce înseamnă că *Vasile* este acum decan), dar care nu au fost egali în trecut (când *Vasile* nu era decan). Pe baza acestui fapt, din

(TRECUT (VIZITA1 VASILE1 FRANTA1)),
nu se poate decide dacă decanul din trecut a vizitat Franţa, adică
(TRECUT (VIZITA1 DECAN1 FRANTA1)).

De remarcat că o propoziţie şi negaţia ei pot fi ambele adevărate în trecut (dar la momente diferite). De exemplu, propoziţiile

Aurel a fost vesel.
Aurel nu a fost vesel.

pot fi ambele adevărate la trecut, deci

(TRECUT (VESEL AUREL1)),
(TRECUT (NOT ((VESEL AUREL1)))

pot fi simultan adevărate.

În continuare vom ignora totuşi efectul operatorilor modali de timp.

9.4 Codificarea ambiguităţii în forma logică

După cum se ştie, o propoziţie tipică poate avea mai multe structuri sintactice posibile, fiecare din ele cu multiple forme logice. În plus, cuvintele propoziţiei pot avea mai multe sensuri. Simpla enumerare a formelor logice posibile nu este deci practică. Este mai eficient să construim o cale prin care anumite ambiguităţi comune pot fi unificate şi reprezentate local într-o formă logică. În unele lucrări această manieră de codificare a ambiguităţilor este tratată ca un nivel separat de reprezentare în forma logică, numit *forma cvasi-logică.*

Poate cea mai mare sursă de ambiguitate în forma logică provine din faptul că multe cuvinte au sensuri multiple. Unele sensuri au proprietăţi structurale diferite, astfel că ele pot fi eliminate de contextul în care se află scufundată propoziţia. În prezent, singurul mod de a codifica acestea va fi construcţia unei forme logice separate pentru fiecare combinaţie posibilă de sensuri ale cuvintelor din propoziţie. Pentru a reduce această explozie a formelor logice, putem folosi aceeaşi tehnică ca la asocierea de valori multiple de caracteristici, cunoscută la structura sintactică. Anume, apariţia într-o expresie a unui sens atomic, se poate înlocui cu un set de sensuri atomice.

Exemplul 9.12 *Substantivul "cheie" are cel puţin două sensuri: CHEIE1 - obiectul necesar pentru deschiderea uşilor, şi CHEIE2 - metoda de decriptare folosită de destinatarul unui mesaj cifrat. Deci, propoziţia*

Elena a folosit cheia.

utilizată în afară de context, este ambiguă. Ambele sensuri posibile pot fi reprezentate printr-o singură formă logică; anume:

(ART_A **b1***:({CHEIE1 CHEIE2}* **b1***)(TRECUT (FOLOSI1 ELENA* **b1***))).*

Această scriere este obţinută din cele două forme logice posibile:

- *(ART_A* **b1***:(CHEIE1* **b1***)(TRECUT (FOLOSI1 ELENA* **b1***)))*

- *(ART_A* b1:*(CHEIE2* b1*)(TRECUT (FOLOSI1 ELENA* b1*)))*

Una din cele mai complexe forme de ambiguitate provine din domeniul relativ de acţiune al cuantificatorilor şi operatorilor. Am văzut că o propoziţie cum ar fi

Orice băiat iubeşte un câine.

este ambiguă între cele două tipuri de citiri, după cum este definit domeniul cuantificatorilor. Nu există nici o metodă independentă de context pentru a rezolva această ambiguitate, aşa că ambele citiri vor trebui reprezentate în formele finale logice ale propoziţiei. Dar, în loc de a enumera toate domeniile posibile - ceea ce ar duce la o creştere exponenţială a interpetărilor bazate pe numărul de construcţii de domenii - vom introduce o prescurtare în limbajul formei logice, care reduce masiv interpretările. Pur şi simplu forma logică abreviată nu va conţine informaţia relativă la domeniu. În plus, construcţii specifice (cum ar fi cuantificatorii generalizaţi) vor fi tratate sintactic ca termeni şi vor apare în poziţia indicată de structura sintactică a propoziţiei. Ele sunt marcate prin paranteze unghiulare pentru a indica domeniul de abreviere.

Exemplul 9.13 *Formele logice pentru*

Orice băiat iubeşte un câine.

sunt prinse într-o singură formă ambiguă:
(IUBEŞTE1 < ∀ **b1** *(BĂIAT1* b1*)*>< ∃ **d1** *(CÂINE1* **d1***)*>*)*

Această prescurtare este o ambiguitate între formele logice

(∀ b1:*(BĂIAT1* b1*)(*∃ d1 *(CÂINE1* d1*)(IUBEŞTE1* **b1 d1***)))*
(∃ d1:*(CÂINE1* d1*)(*∀ b1:*(BĂIAT1* b1*)(IUBEŞTE1* **b1 d1***)))*

Dacă restricţia într-un cuantificator generalizat este o propoziţie care utilizează un singur predicat unar, se poate folosi o abreviere care să elimine variabila. De exemplu, forma

< ∀ **b1** *(BĂIAT* b1*)*>
poate fi scrisă
< ∀ **b1** *BĂIAT*>.

Multe construcţii din limbajul natural sunt senzitive la domenii. În particular, toţi cuantificatorii generalizaţi, inclusiv articolele hotărâte. De exemplu, pentru propoziţia

La orice hotel recepţionistul era amabil.

în aproape orice context *recepţionistul* intră în domeniul lui *orice hotel*, adică există câte un recepţionist diferit la fiecare hotel.

De asemenea operatori cum ar fi negaţia sau timpul sunt senzitivi la domeniu. Folosind un exemplu din limba engleză, propoziţia

Every boy didn't run.

este ambiguă între cele două tipuri de citiri (distributivă şi colectivă): astfel, fie doar unii băieţi nu au alergat, adică:

(NOT (EVERY **b1***:(BOY1* **b1***)(RUN1* **b1***)))*

fie nici un băiat nu a alergat:

(EVERY **b1***:(BOY1* **b1***)(NOT (RUN1* **b1***)))*

Ambele citiri pot fi scrise sub o singură formă logică:

(<NOT RUN1><EVERY **b1** *BOY1>)*

unde operatorii unari fără domeniu (*NOT, PAST, PRES,...*) au fost omişi.

Ca o ultimă observaţie, să comentăm ce se întâmplă cu numele proprii şi cu pronumele. Vom presupune că fiecare nume propriu identifică un sens caracteristic unui obiect din domeniu. Orice nume propriu trebuie interpretat în context; astfel, numele *Andrei* se va referi la persoane diferite în situaţii diferite. În semantică vom introduce această construcţie ca o funcţie specială de forma:

(NUME < variabilă >< nume >)

care are ca efect producerea în contextul curent a unui obiect cu numele respectiv.

Exemplul 9.14 *Deci forma cvasi-logică pentru propoziţia*

Andrei a alergat.

este

(<TRECUT ALERGA1>(NUME **j1** *"Andrei")).*

Un argument similar poate fi dat şi pentru pronume sau alte cuvinte indexate ca de exemplu *aici, ieri* etc. Acestea vor fi prelucrate folosind funcţii speciale de forma

(PRO < variabilă >< propoziţie >).

Exemplul 9.15 *Forma cvasi-logică pentru propoziţia*

Toţi oamenii l-au iubit.

este:

(<TRECUT IUBI1><PLURAL (TOT **m1** *OM1)>(PRO* **m2** *(EL1* **m2***)))*

EL1 *este acelaşi pentru* **el** *şi* **lui** *şi - formal - este un predicat cu valoare de adevăr pentru obiectele care satisfac restricţiile precedente.*

Ca şi cuantificatorii generalizaţi, când restricţiile sunt predicate unare, formele pronumelor pot fi abreviate. De exemplu, forma logică pentru *el* poate fi scrisă adesea *(PRO* m2 *EL1)*.

Construcţiile descrise aici reduc semnificativ numărul formelor logice. Totuşi, nu toate ambiguităţile pot fi modelate prin prescurtări, aşa că vor fi propoziţii care vor necesita liste de forme logice posibile, chiar şi pentru o singură structură sintactică.

Prelegerea 10

Modele şi structuri semantice

10.1 Verbe şi stări în formă logică

Să considerăm verbele acţionând ca predicate în forma logică. În acest fel se pot prelucra toate reprezentările lor distincte şi se pot folosi diverse proprietăţi de adnotare, pierzându-se doar câteva generalităţi.

Exemplul 10.1 *Să considerăm următoarele afirmaţii, toate folosind verbul "a sparge":*

1. *Radu a spart geamul cu ciocanul.*

2. *Ciocanul a spart geamul.*

3. *Geamul s-a spart.*

Intuitiv, toate aceste afirmaţii descriu acelaşi tip de eveniment, în detalii diferite. Deci ar fi de dorit ca verbul *a sparge* să corespundă aceluiaşi sens în fiecare caz. Problema care apare constă în faptul că în cele trei afirmaţii, verbul este folosit cu arităţi diferite. În prima propoziţie *a sparge* indică o relaţie ternară între ***Radu***, *geam* şi *ciocan*, a doua propoziţie este o relaţie binară între *ciocan* şi *geam*, iar a treia propoziţie este o relaţie unară relativă la *geam*. Deci se pare că sunt necesare trei sensuri distincte ale verbului *a sparge*: *SPARGE1, SPARGE2, SPARGE3*, care diferă prin aritate şi produc forme logice de tipul:

1. *(<TRECUT SPARGE1>(NUME* j1 *"Radu")*
   <ART_UL w1 *GEAM1><ART_UL* h1 *CIOCAN1>)*,

2. *(<TRECUT SPARGE2><ART_UL* h1 *CIOCAN1>)*
   <ART_UL w1 *GEAM1>)*

3. *(<TRECUT SPARGE3><ART_UL* w1 *GEAM1>)*

Pentru a garanta că toate predicatele sunt interpretate corespunzător, fiecare reprezentare va trebui completată cu axiome care să asigure echivalenţa cu celelalte două forme. Aceste axiome sunt numite *postulate de semnificaţie*.

Pare însă destul de neconvenabil să specificăm astfel de restricţii pentru fiecare verb. O variantă de rezolvare (Davidson, 1967) constă în introducerea drept variabile a evenimentelor. Acum, înţelesul unui propoziţii de tipul

Radu l-a spart.

poate fi scris:

(∃ **e1**: *(SPARGE* **e1** *(NUME* **j1** *"Radu")(PRO* **i1** *L)))*

(s-a omis predicatul de timp) care semnifică faptul că *e1* este un eveniment în care *Radu* sparge un anumit geam.

În mod similar, înţelesul propoziţiei

Radu l-a spart cu ciocanul.

este:

(∃ **e1**: *(& (SPARGE* **e1** *(NUME* **j1** *"Radu")(PRO* **i1** *L))*
(INSTR **e1** *<ART_L* **h1** *CIOCAN>)))*

Avantajul constă în faptul că modificatorii suplimentari, cum ar fi *cu ciocanul*, sau *din greşeală*, sau *marţea trecută* pot fi adăugaţi la reprezentarea de bază cu predicate care utilizează evenimentul respectiv.

Motivări similare au condus ulterior (începutul anilor '70) la dezvoltarea gramaticilor de caz, care au introdus idei folosite şi acum. Una din ele este aceea că există un set limitat de relaţii semantice care pot fi stabilite între un verb şi argumentele sale. Aceste relaţii sunt numite *roluri tematice* sau *roluri de caz*; deşi au fost folosite în multe aplicaţii practice, numărul lor a rămas totdeauna restrâns.

Revenind la exemplul de sus, ideea este aceea că *Radu, ciocanul* şi *geamul* joacă în toate cele trei propoziţii aceleaşi roluri semantice. *Radu* este factorul activ, actorul (rolul *agent*), *geamul* este obiectul (rolul *temă*) iar *ciocanul* este instrumentul (rolul *instrument*) folosit în actul de spargere. Aceasta sugerează o reprezentare a înţelesului propoziţiei, similară cu cea folosită în reţelele semantice, unde totul este exprimat în funcţie de relaţii unare şi binare.

De exemplu, pe baza celor trei roluri tematice evidenţiate mai sus, înţelesul propoziţiei

Radu a spart geamul.

va fi:

(∃ **e** *(& SPARGE* **e***)(AGENT* **e** *(NUME* **j1** *"Radu"))*
(TEMA **e** *<ART_UL* **w1** *GEAM>)))*

Deoarece aceste construcţii sunt foarte des folosite, vom introduce pentru ele o nouă notaţie. Forma prescurtată pentru o afirmaţie de forma

$$(\exists \mathbf{e} : (\&(\textit{Eveniment-p}\ \mathbf{e})(\textit{Relaţie}_1\ \mathbf{e}\ \textit{obj}_1)\ldots(\textit{Relaţie}_n\ \mathbf{e}\ \textit{obj}_n)))$$

este:

$$(\textit{Eveniment-p}\ \mathbf{e}\ [\textit{Relaţie}_1\ \textit{obj}_1]\ldots[\textit{Relaţie}_n\ \textit{obj}_n])$$

În particular, cu această abreviere, forma cvasi-logică pentru propoziţia de sus este:

(<TRECUT SPARGE1> e1 [AGENT (NUME j1 "Radu")]
[TEMA <ART_UL w1 GEAM1>])

Argumente similare pot fi evident construite şi pentru verbe. Astfel, să considerăm propoziţia

Maria a fost nefericită.

Dacă ea este reprezentată (folosind un predicat unar) sub forma

(<TRECUT NEFERICIT>(NUME j1 "Maria"))

să vedem cum poate fi folosită această informaţie pentru propoziţia

Maria a fost nefericită la întâlnire.

În această situaţie vom generaliza noţiunea de eveniment, pentru a include stări, după care se foloseşte aceeaşi tehnică. De exemplu, se poate considera *NEFERICIT* drept predicat care are ca argument o stare de *nefericire*, şi se foloseşte rolul *TEMA* pentru a include pe *Maria*:

(∃ s <TRECUT NEFERICIT> s)(TEMA s (NUME j1 "Maria"))

Folosind aceleaşi convenţii de prescurtare (definite la evenimente), propoziţia de sus se poate reprezenta sub forma

(<TRECUT NEFERICIT> s [TEMA (NUME j1 "Maria")]
[LA_LOC m1 ÎNTÂLNIRE])

În general - în funcţie de situaţie - se alege unul din cele două moduri de reprezentare a înţelesului propoziţiilor. Astfel, semnificaţia propoziţiei

Sanda vede Bucegii.

poate fi dată atât de
(PREZENT (VEDE1 x1 [AGENT (NUME j1 "Sanda")]
[TEMA (NUME m1 (ART_I "Bucegi"))]))
care este echivalent cu
(PREZENT (∃ x1 (& VEDE1 x1)(AGENT x1 (NUME j1 "Sanda"))
(TEMA x1 (NUME m1 (ART_I "Bucegi"))))))
dar şi sub forma predicatelor argument:
(PREZENT (VEDE1 (NUME j1 "Sanda")(NUME m1 (ART_I "Bucegi")))).

10.2 Rolurile tematice

Să trecem în revistă mai detaliat teoriile bazate pe noţiunea de *roluri tematice* (sau *cazuri*). Vom porni de la cele trei propoziţii date în exemplul 10.1, în care *Radu*, *ciocanul* şi *geamul* joacă aceleaşi roluri semantice: *Radu* este *actorul*, *geamul* este *obiectul* iar *ciocanul* este *instrumentul* folosit în actul de spargere al geamului. Pentru a reţine aceste idei în forma cvasi-logică, introducem relaţiile *AGENT*, *TEMA* şi *INSTR*.

Poate că rolul tematic cel mai uşor de definit este rolul *AGENT*.

Definiţia 10.1 *O expresie substantiv îndeplineşte rolul AGENT dacă desemnează instigatorul acţiunii descrise de propoziţie.*

Acest rol poate atribui voinţă, intenţie sau responsabilitate pentru acţiunea descrisă de agent. Un test pentru verificarea alegerii corecte a agentului este acela de a introduce în propoziţii expresii cum ar fi *intenţionat* sau *trebuie*, care să întărească rolul celui presupus a fi *AGENT*; dacă propoziţia obţinută este corect formată, expresia substantiv aleasă poate avea rolul de *AGENT*.

De exemplu, următoarele propoziţii sunt acceptabile:

Radu, intenţionat a spart geamul cu ciocanul.
Radu a trebuit să spargă geamul cu ciocanul pentru a avea mai mult aer.

În schimb următoarea propoziţie nu este acceptabilă (pragmatic):

Ciocanul, intenţionat a spart geamul.

Referitor la a treia propoziţie, dacă scriem însă:

Geamul a trebuit să se spargă pentru a avea mai mult aer.

nu putem spune că ea nu este corectă. Dar acum propoziţia se referă la o altă persoană, (*Radu*), nespecificată în propoziţie, care trebuie să aibă mai mult aer. Deci, peste tot *Radu* joacă rol de *AGENT*.

Ideea de a detecta agentul prin metoda descrisă mai sus nu este totdeauna posibilă. De exemplu, nu vom putea afirma în mod normal:

Radu, intenţionat a murit.
Paula a trebuit să-şi amintească de ziua ei de naştere, pentru a primi cadouri.

Bineînţeles, se pot construi propoziţii echivalente cu acestea (de exemplu, pentru prima se poate spune *Radu s-a sinucis*), dar astfel se modifică sensul iniţial al propoziţiei (*Radu a murit*).

Definiţia 10.2 *Expresiile substantivale care suferă anumite schimbări sau asupra cărora se acţionează, îndeplinesc rolul de TEMĂ.*

Aceasta corespunde de obicei entităţii sintactice *OBIECT* şi, pentru un anumit verb tranzitiv X, constituie răspunsul la întrebarea

Ce a fost X-at ?

Exemplul 10.2 *Pentru propoziţia*
Vulturul a văzut şoarecele.
NP "şoarecele" este TEMA şi răspunsul la întrebarea

"Ce a fost văzut ?"

Pentru verbele intranzitive rolul *TEMĂ* este îndeplinit de expresia substantivală subiect care nu este *AGENT*. Deci, în

Norii apăreau deasupra orizontului.

expresia substantivală *norii* va fi *TEMA*.

Exemplul 10.3 *Alte exemple:*
Piatra s-a spart.
Vasile a spart piatra.
I-am dat lui Costel cartea.
(TEMA a fost subliniată).

Pentru locaţii şi adresări se definesc alte roluri tematice. Aici va trebuie să distingem din nou între relaţiile care indică un loc (adresă) şi cele care indică o adresare (drum).

Astfel, relaţia *LA_LOC* indică poziţia unui obiect sau locul unde se petrece un eveniment, ca în exemplele:

Silviu se plimbă pe drum.
Scaunul este lângă uşă.

Pe drum descrie unde are loc plimbarea, iar *lângă uşă* indică localizarea scaunului.

Alte propoziţii descriu schimbarea locului, direcţia de mişcare, sau drumul:

Ieri m-am plimbat de aici până la şcoală.
Cădea spre pământ.
Păsările zburau dinspre lac de-a lungul râului.

Există aici cel puţin trei tipuri diferite de propoziţii: cele care descriu unde-vine-cineva-de-undeva (rolul *DIN_LOC*) - cazul lui de *aici*; cele care descriu destinaţia (rolul *SPRE_LOC*) - ca în spre *pământ*; şi cele care descriu traiectoria sau drumul (rolul *DRUM_LOC*) - cum este *de-a lungul râului*.

Aceste roluri de adresă pot fi generalizate în roluri peste valori arbitrare de stări (numite rol *LA*) şi roluri pentru schimbarea unei stări arbitrare (roluri *DIN, SPRE, DRUM*).

Deci, *LA_LOC* este o particularizare a rolului *LA* etc. Alte particularizări ale acestui rol se pot obţine plecând de la relaţia abstractă de posesie. Astfel:

Exemplul 10.4

I-am aruncat mingea lui Tudor.	*(rolul SPRE_LOC)*
I-am dat cartea lui Tudor.	*(rolul SPRE_POSS)*
Am primit mingea de la Tudor.	*(rolul DIN_LOC)*
Am împrumutat cartea de la Tudor.	*(rolul DIN_POSS)*
Cutia conţine o minge.	*(rolul LA_LOC)*
Tudor posedă o carte.	*(rolul LA_POSS)*

Similar, se pot defini rolurile *LA_TIMP*, *SPRE_TIMP*, *DIN_TIMP*, ca în:

Exemplul 10.5

Am văzut maşina la ora 3.	*(rolul LA_TIMP)*
Am lucrat de la unu la trei.	*(rolul DIN_TIMP şi SPRE_TIMP)*

Sau roluri relative la valoare, ca în:

Exemplul 10.6

Temperatura rămâne la zero.	*(LA_VALOARE)*
Temperatura creşte peste zero.	*(DIN_VALOARE)*

Un alt rol este motivat de faptul că, în aceste coordonate, nu se poate clasifica uşor rolul *NP* dintr-o propoziţie cum ar fi:

Radu a crezut că plouase.

Rolul *TEMĂ* este îndeplinit de clauza *că plouase*, deoarece aceasta este ceea ce credea subiectul. *Radu* nu poate fi *AGENT* deoarece nu era intenţia lui să creadă ceva. Deci va trebui introdus un nou rol, numit *EXPERIENŢĂ*, îndeplinit de fiinţe aflate într-o anumită stare psihică, sau care îndeplinesc o anumită stare psihică (cum ar fi percepţia - în exemplul anterior).

Alt rol este cel de *BENEFICIAR*, îndeplinit de persoane pentru care este realizat un anumit eveniment, ca în:

Am cumpărat flori pentru Lucia.
Găseşte-mi hârtiile !
I-am dat cartea lui Mihai pentru Tamara.

Ultimul exemplu arată necesitatea de a distinge între rolul *SPRE_POSS* (adică *lui Mihai*) şi *BENEFICIAR*.

Rolul *INSTR* descrie un mecanism - material sau forţă - folosite pentru a realiza un anumit eveniment.

Henry a spart oglinda cu bastonul.
Bastonul a spart oglinda.
Am folosit ceva mirodenii pentru a face o prăjitură.
Am făcut o prăjitură cu ceva mirodenii.

În funcţie de verb, rolul *INSTR* poate fi folosit uneori ca subiect, atunci când rolul *AGENT* nu este specificat. Forţele naturale sunt de asemenea incluse în categoria *INSTR*, ca în:

Soarele a ars culturile.
Valentin a folosit soarele pentru a usca pânza.

Rolurile *AGENT* şi *INSTR* pot fi combinate într-un rol mai general numit *AGENT_CAUZAL*.

Alte roluri trebuiesc identificate înainte de a începe analiza anumitor propoziţii. De exemplu, unele afirmaţii descriu situaţii în care doi oameni realizează împreună o acţiune; să considerăm propoziţia:

Paul a mutat dulapul ajutat de Virgil.

Pentru a o prelucra, trebuie definit un rol nou - *CO_AGENT* pentru introducerea expresiei prepoziţionale *ajutat de Virgil*.

Un caz mai complicat apare în propoziţiile care descriu schimburi sau alte interacţiuni complexe. Să considerăm exemplele:

Ion a plătit anticarului 10.000 *lei pentru carte.*
Ion a luat cartea de la anticar pentru 10.000 *lei.*

Ambele propoziţii descriu o situaţie în care *Ion* a dat 10.000 lei unui anticar şi

a primit în schimb o carte. În prima propoziţie 10.000 lei este *TEMA*, iar *cartea* nu are asignat nici un rol. În a doua propoziţie, situaţia este inversată: *cartea* este *TEMA* iar 10.000 lei nu are atribuit nici un rol. Pentru prelucrarea acestor cazuri se defineşte un rol *CO_TEMĂ* pentru al doilea obiect aflat în operaţia de schimb.

Problema poate fi generalizată considerând propoziţii care descriu două evenimente. **Evenimentul primar** este cel focalizat până acum; pe lângă el poate apare şi un **eveniment secundar**. În această modalitate, prima propoziţie din exemplu poate fi analizată astfel (s-a considerat drept eveniment primar *Ion plătind suma de 10.000 lei*, iar ca eveniment secundar *Ion primind cartea*):

Expresie	Primar	Secundar
Ion	*AGENT*	*AGENT*
10.000 lei	*TEMA*	–
anticar	*SPRE_POSS*	*DIN_POSS*
cartea	–	*TEMA*

După cum s-a văzut, verbele pot fi clasificate după rolurile tematice pe care le solicită. Pentru a face o clasificare mai precisă, va trebui să punem în evidenţă rolurile legate intrisec de verbe. De exemplu, aproape orice verb la timpul trecut are un rol *LA_TIMP* realizat de un adverb (cum ar fi *ieri*). Deci acest rol pare a fi o proprietate mai strâns legată în general de expresiile verbale, decât o proprietate a oricărui verb individual. Mai pot apare şi alte astfel de roluri, anume cele realizate de constituienţii pentru care verbele au definite subcategorii.

De exemplu, verbul *a pune* are definită ca subcategorie un *PP* şi deci acest *PP* trebuie să realizeze un rol *SPRE_LOC*.

În clasificarea verbelor, aceste tipuri de roluri sunt importante; ele se numesc **roluri interne** ale verbelor.

Această importanţă sugerează necesitatea unui test pentru a determina dacă un anumit rol este intern sau nu pentru un verb dat. Folosind exemplele de sus, acest test se poate formula astfel:

Dacă rolul este obligatoriu, atunci este un rol intern.

Există totuşi roluri interne care apar ca opţionale în anumite situaţii; pentru ele sunt necesare alte teste.

Se poate construi un test bazat pe observaţia că orice verb are în rolurile interne cel mult un *NP*. Dacă sunt necesare mai multe *NP*-uri, acestea sunt legate printr-o conjuncţie. Astfel, se poate spune

Andrei şi cu mine am alergat la fereastră.

dar nu

Andrei mine am alergat la fereastră.

La fel,

Am alergat la magazin şi la bancă.

este corect, pe când

Am alergat la magazin la bancă.

nu este. Deci rolurile *AGENT* şi *SPRE_LOC* sunt interne pentru verbul *a alerga*.

Verbele au până la trei roluri interne, cel puţin unul din ele fiind realizat în propoziţia care foloseşte verbul. Uneori trebuie să existe obligatoriu un anumit rol (de exemplu *SPRE_LOC* pentru verbul *a pune*). De asemenea, *TEMA* este obligatorie, în timp ce *AGENTUL* este totdeauna opţional pentru verbele folosite la diateza pasivă. Există şi restricţii sintactice referitoare la modul de realizare al diverselor roluri. În Tabelul 10.1 sunt listate câteva modalităţi simple de realizare a rolurilor în propoziţii.

Tabelul 10.1:

Rol	**Realizare**
AGENT	Ca subiect în propoziţii active
	După prepoziţia *prin* în propoziţii pasive
TEMA	Ca obiect al verbelor tranzitive
	Ca subiect al verbelor non-active
INSTR	Ca subiect în propoziţiile active fără agent
	După prepoziţia *cu*
EXPERIENŢĂ	Ca subiect în propoziţii active fără agent
BENEFICIAR	Ca obiect indirect cu verbe tranzitive
	După prepoziţia *pentru*
LA_LOC	După prepoziţiile *in, pe, după*, ...
LA_POSS	Expresii substantivale posesive
	Ca subiect al propoziţiilor, dacă nu este agent
SPRE_LOC	După prepoziţiile *la, spre*, ...
SPRE_POSS	După prepoziţia *la*, obiect indirect cu anumite verbe
DIN_LOC	După prepoziţiile *din, afară*, ...
DIN_POSS	După prepoziţia *din*

Exemplul 10.7 *Să mai dăm câteva exemple; verbele sunt subliniate iar rolurile (asigurate de NP,PP sau S) sunt listate în ordinea apariţiei lor în frază.*

Ion a alergat.	*AGENT*
Ion a alergat cu un buchet de flori.	*AGENT + INSTR*
Ion a alergat cu un buchet de flori pentru Ana.	*AGENT + INSTR+ + BENEFICIAR*
Ion a stricat maşina.	*AGENT + TEMA*
Ion a dat ziarul prin fereastră.	*AGENT + TEMA + DRUM*
Ion i-a plătit lui Barbu maşina.	*AGENT + SPRE_POSS + TEMA*
Barbu a împins maşina de la casa lui Ion până la atelier.	*AGENT + TEMA+ + DIN_LOC + SPRE_LOC*
Ion este înalt.	*TEMA*
Barbu crede că Ion este înalt.	*EXPERIENŢĂ + TEMA*
Ana are o maşină.	*LA_POSS + TEMA*
Eu sunt în clasă.	*TEMA + LA_LOC*
Mingea s-a rostogolit jos, de pe mal în apă.	*TEMA + DRUM + SPRE_LOC*

10.3 Exprimarea orală şi propoziţii derivate

Procesul de comunicare constă din exprimarea unor propoziţii; acestea sunt folosite în diverse scopuri. Fiecare exprimare a unei propoziţii indică o relaţie între vorbitor şi conţinutul propoziţional al pronunţiei.

Un prim pas în surprinderea acestor afinităţi constă în extinderea formei logice. Astfel, fiecare tip principal de propoziţie are un operator ataşat care ia interpretarea propoziţiei drept argument şi produce ceea ce se numeşte ***actul de suprafaţă*** al vorbirii. Acesta indică cum se intenţionează folosirea propoziţiei alese în actualizarea situaţiei discursului.

Operatorii utilizaţi în acest scop sunt:

ASSERT	:	propoziţia este de tip aserţiune (enunţiativă);
Y/N-QUERY	:	propoziţia este interogativă;
COMMAND	:	propoziţia descrie o acţiune care trebuie realizată;
WH-QUERY	:	propoziţia descrie un obiect care trebuie identificat.

Pentru propoziţii enunţiative, de tipul

Omul a mâncat un măr.

forma logică completă este:

(ASSERT(<TRECUT MÂNCA> **e1** *[AGENT <ART_UL m1 OM1>]*
[TEMA <UN p1 MĂR1>]))

Pentru propoziţii interogative (cu răspuns *da/nu*), cum ar fi

A mâncat omul un măr ?

forma logică completă este:

(Y/N-QUERY (<TRECUT MÂNCA> e1 [AGENT <ART_UL **m1** *OM1>]*
[TEMA <UN p1 MĂR1>]))

Pentru comenzi, cum ar fi

Mănâncă mărul !

forma logică este:

(COMMAND (MÂNCA **e1** *[TEMA <ART_UL* **p1** *MĂR>]))*

Pentru întrebări de tipul

Ce a mâncat omul ?

trebuiesc introduse elemente noi în limbajul formal. În primul rând este necesară stabilirea unui mod de reprezentare a înţelesului expresiilor substantivale care folosesc termeni de forma *ce (what)*; aceasta se realizează prin cuantificatorul *CE* care indică faptul că termenul se referă la obiectul (sau obiectele) despre care se întreabă. Ca o remarcă, aceşti *ce*-termeni sunt senzitivi de domeniu, aşa că este posibilă folosirea lor pe rol de cuantificatori.

Deci pentru expresiile substantivale interogative simple avem reprezentările:

ce?	< *CE* **p1** NIMIC >
care om?	< *CE* **m1** OM1 >
cine?	< *CE* **p1** PERSOANĂ >

Cu aceste definiţii, forma logică pentru propoziţia interogativă de sus este:

(WH-QUERY (<TRECUT MÂNCA> **e1** *[AGENT <ART_UL* **m1** *OM1>]*
[TEMA <CE **w1** *OBIECT_FIZIC>]))*

Pentru expresii interogative relative la cantitate, cum ar fi *câte* sau *cât de mult* se definesc cuantificatorii *CÂT* respectiv *CÂT_DE_MULT*.

Propoziţiile derivate, cum ar fi clauzele relative, completează la sfârşit expresiile substantivale cu restricţii, deci nu vor necesita notaţii noi.

Exemplul 10.8 *Forma logică a propoziţiei*
Omul care a mâncat un măr a plecat.
este:
(ASSERT
(<TRECUT PLECA> **k1**
[AGENT <ART_UL **m1** *(& (OM1* **m1***)*
(<TRECUT MÂNCA1> **e2**
[AGENT **m1***]*
[TEMA <UN **p1** *MĂR>]))>]))*

În Tabelul 10.2 se află o definiţie formală a sintaxei şi limbajului pentru formele logice discutate aici, în cazul limbii engleze.

Tabelul 10.2:

PRONUNŢIE	⟶	(ASSERT PROP)|
		(Y/N-QUERY PROP)|
		(COMMAND PROP)|
		(WH-QUERY PROP)
PROP	⟶	(n – AR_OPER PROP$_1$ ... PROP$_n$)|
		(QUANTIF VAR: PROP PROP)|
		(n – AR_PRED TERM$_1$ ... $TERM_n$)|
		(EV_ST_PRED VAR [ROL TERM]$_1$ ... [ROL TERM]$_n$)
TERM	⟶	VAR|
		(NUME_VAR NUME_ŞIR)|
		(PRO VAR PROP)
1 – AR_OPER	⟶	*NOT|PAST|PERF|PROG|...*
2 – AR_OPER	⟶	*AND|BUT|IF – THEN|...*
QUANTIF	⟶	*THE|SOME|WH|∃|...*
VAR	⟶	**b1—man3**|...
1 – AR_PRED	⟶	*TIP_PRED|HAPPY1|RED1|...*
TIP_PRED	⟶	*EV_ST_PRED|(PLURTIP_PRED)|MAN1|...*
EV_ST_PRED	⟶	*RUN1|LOVE3|GIVE1|HAPPY|...*
2 – AR_PRED	⟶	*ROL|ABOVE|...*
ROL	⟶	AGENT|TEMĂ|*LA_LOC|INSTR|...*
NUME_ŞIR	⟶	"John"|"The New York Times"|...

Prelegerea 11

Semantică şi teoria modelelor

11.1 Teoria modelelor în semantică

În general, termenul de semantică reflectă reprezentarea înţelesului propoziţiilor; aceasta a fost baza limbajului bazat pe logica formală.

Mai există însă un înţeles al termenului de semantică, folosit în teoria limbajelor formale, care conferă la rândul lui un înţeles limbajului logicii formale. El se concentrează pe distingerea diferitelor clase de obiecte semantice, explorând proprietăţile modelului teoretic, adică definind unităţi semantice în termenii aplicaţiilor din teoria mulţimilor.

Construcţia de bază pentru definirea proprietăţilor semantice este aceea de *model*, care intuitiv constă dintr-o mulţime de obiecte, împreună cu proprietăţile şi relaţiile lor, însoţite de specificarea legăturii dintre limbajul studiat şi aceste relaţii. Un model poate fi gândit ca reprezentarea unui context particular în care este evaluată o propoziţie. Se pot impune diverse restricţii care să reprezinte anumite proprietăţi specifice unui anumit model. De exemplu, modelele standard pentru logică, numite *modele Tarskiene*, sunt complete, în sensul că ele asociază fiecărui termen din limbaj un obiect al modelului şi fiecărei propoziţii o valoare de adevăr (*true/false*). Există însă diverse forme de modele parţiale în care nu se impune ca orice propoziţie să fie sau adevărată sau falsă. Astfel de modele sunt mai apropiate de situaţiile descrise în prelegerea anterioară şi ele pot fi baza unei teorii matematice a situaţiilor, care să constituie un model formal.

Teoria modelelor constituie o metodă adecvată pentru studiul înţelesului independent de context, pentru că înţelesul propoziţiilor nu este definit în raport cu un anumit model, ci de felul cum se leagă ele de un model abstract. Altfel spus, înţelesul unei propoziţii este definit în termenii proprietăţilor pe care le are acesta în raport cu un model arbitrar.

Definiţia 11.1 *Se numeşte model un cuplu $m = < D_m, I_m >$, unde D_m (domeniul de interpretare) este format dintr-o mulţime finită şi nevidă de "primitive", iar $I_m : S \longrightarrow 2^{D_m}$ este funcţia de interpretare. (S este un set de obiecte al sistemului modelat).*

Pentru a modela limbajul natural, domeniul de interpretare trebuie să aibă obiecte pentru toate tipurile distincte de elemente la care se poate face referire

(obiecte fizice, timpi, locaţii, evenimente, situaţii). Funcţia de interpretare aplică sensuri şi structuri mai largi în structuri definite în domeniu.

Exemplul 11.1 *Un tip de funcţie de interpretare pentru sensurile următoarelor clase lexicale, poate fi:*

- *Sensuri ale expresiilor substantivale - se referă la obiecte specificate; funcţia de interpretare asociază fiecăruia câte un element din D_m.*
- *Sensuri ale substantivelor comune la singular (cum ar fi "câine", "petrecere", "idee") - identifică clase de obiecte din domeniu; funcţia de interpretare le aplică în submulţimi din D_m.*
- *Sensuri ale verbelor - identifică mulţimi de relaţii n-are între obiecte din D_m. Aritatea fiecărei relaţii depinde de verb. De exemplu, pentru verbul "a alerga", ALERGA1 are valori într-un set de relaţii unare ($< X >$ unde "X aleargă"); pentru "a iubi", IUBEŞTE1 are valori într-un set de aplicaţii binare ($< X, Y >$ unde "X iubeşte Y"); pentru "a pune", PUNE1 are valori într-un set de relaţii ternare ($< X, Y, Z >$, unde "X pune Y în locaţia Z").*

Se poate defini acum noţiunea de *adevăr* a unei propoziţii (reprezentate în limbajul logicii formale) relativ la un model **m**.

Definiţia 11.2 *O propoziţie de forma ($V_n\ a_1a_2\ldots a_n$) este adevărată în raport cu modelul* **m** *dacă şi numai dacă $< Im(a_1), \ldots, Im(a_n) >\in Im(V_n)$.*

În notaţie clasică, aceasta se scrie

$$\mathbf{m} \vdash P$$

şi semnifică faptul că propoziţia P este adevărată în raport cu modelul **m**.
Relaţia se mai citeşte "**m** *este suport pentru P*".

Exemplul 11.2 **m** *este suport pentru (ALERGA1 PAUL1) numai dacă* $< I_m(PAUL1) >\in I_m(ALERGA1)$;

m *este suport pentru (IUBEŞTE1 RADU1 MARIA1) numai dacă* $< I_m(RADU1), I_m(MARIA1) >\in I_m(IUBEŞTE1)$.

Semantica pentru negaţie depinde de proprietăţile modelelor folosite. În semanticile clasice Tarskiene (unde modelele sunt complete) negaţia este definită prin afirmaţia:

m este suport pentru (*NOT P*) dacă şi numai dacă **m** nu este suport pentru P.

În alte modele, condiţia aceasta este prea tare, deoarece pot apare propoziţii care să nu fie nici adevărate şi nici false. Problema se rezolvă definind câte două mulţimi disjuncte pentru fiecare nume de relaţie; fiecare din ele conţine secvenţe pentru care relaţia este adevărată respectiv falsă. Orice secvenţă care nu se află în nici una din cele două mulţimi va fi nici adevărată şi nici falsă.

Semanticile cuantificatorilor din *FOPC* sunt banale. Astfel,

- Propoziţia $\forall x.P(x)$ este adevărată în raport cu un model **m** dacă şi numai dacă $P(x)$ este adevărată pentru orice valoare a lui x din D_m.
- Propoziţia $\exists x.P(x)$ este adevărată în raport cu un model **m** dacă şi numai dacă $P(x)$ este adevărată pentru cel puţin o valoare a lui $x \in D_m$.

Pentru limbajele naturale, clasa cuantificatorilor este mult mai largă. Condiţiile de adevăr pentru fiecare astfel de cuantificator specifică relaţia cerută între obiectele care satisfac cele două aserţiuni de sus.

Exemplul 11.3 *Să considerăm propoziţia (reprezentată în formă logică):*

(MAJORITATEA1 x (P x)(Q x)).

Ea poate fi definită ca adevărată în raport cu un model **m** *dacă şi numai dacă mai mult de jumătate din obiectele din Im(P) sunt în Im(Q). În particular*

(MAJORITATEA1 x (CÂINE1 x)(MUŞCA1 x))

va fi adevărată în raport cu un model **m** *dacă şi numai dacă mai mult de jumătate din elementele mulţimii {x|(CÂINE x)} sunt de asemenea în mulţimea {y|(MUŞCA1 y)} adică mai mult de jumătate din câinii din* **m** *muşcă.*

11.1.1 Definirea relaţiilor semantice pentru propoziţii

Pe baza semanticii astfel definite, se pot stabili relaţii între propoziţii. Anume, dacă se ştie că o propoziţie este adevărată, se poate deduce că şi alte propoziţii sunt adevărate.

Exemplul 11.4 *Dacă se ştie că:*

O cutie roşie este pe masă.

atunci se ştie de asemenea că:

O cutie este pe masă.

O astfel de relaţie între propoziţii se numeşte relaţie de *necesitate*. Referitor la Exemplul 11.4, spunem că propoziţia

O cutie roşie este pe masă.

necesită propoziţia

O cutie este pe masă.

Relaţia de necesitate poate fi definită în termenii modelului care este suport al propoziţiei.

Definiţia 11.3 *Propoziţia S necesită propoziţia S' dacă şi numai dacă pentru orice model* **m***:*

$$m \vdash S \quad \Longrightarrow \quad m \vdash S'.$$

Invers, dacă nu există nici un model suport simultan pentru S şi S', atunci cele două propoziţii nu pot fi adevărate în acelaşi timp (indiferent de situaţie).

Spunem că cele două propoziţii sunt *contradictorii*.

Exemplul 11.5 *Propoziţiile*

O cutie roşie este pe masă.
Pe masă nu este nici o cutie.

sunt contradictorii, pentru că nu există nici un model care să fie suport simultan pentru ambele propoziţii.

Limbajele naturale lucrează de obicei cu o relaţie mai slabă decât cea de necesitate, anume *implicaţia.*

Definiţia 11.4 *O propoziţie S implică S' dacă ori de câte ori S este adevărată, S' poate fi adevărată.*

Exemplul 11.6 *Propoziţia*

Am obosit să merg la facultate în fiecare zi.

implică

Nu mai merg la facultate.

dar nu o necesită. Dacă spunem

Am obosit să merg la facultate în fiecare zi.
O mai fac încă !

va exista un model care să le asigneze ambelor propoziţii valoarea de adevăr; într-un astfel de model, propoziţia

Nu mai merg la facultate.

este falsă.

Una din cele mai dificile probleme în înţelegerea limbajului natural rezultă din faptul că inferenţele folosite sunt mai mult implicaţii decât necesităţi; deci concluziile care se trag la un moment pot fi ulterior reconsiderate şi modificate.

11.1.2 Operatori modali şi semantici ale universurilor posibile

Pentru definirea unei semantici care să utilizeze operatorii modali, trebuiesc introduse structuri noi pe modele. În particular, adevărul unei proprietăţi va fi definit în raport cu o mulţime de cel puţin două modele. Această mulţime poate avea o structură specifică - numită de obicei *structură de model.*

Ca un exemplu, să considerăm operatorul *TRECUT* şi să presupunem că fiecare model individual $\mathbf{m}$ reprezintă starea universului într-un anumit punct în timp. Atunci o structură de model va fi mulţimea modelelor care descriu o istorie posibilă a universului, în care modelele sunt legate unul de altul prin operatorul $<$, unde $\mathbf{m}_1 < \mathbf{m}_2$ semnifică faptul că $\mathbf{m}_1$ descrie o stare a universului anterioară stării descrise de $\mathbf{m}_2$. Adevărul unei propoziţii este acum definit în raport cu o structură de model arbitrară Ω care are între modele sale relaţia de ordine $<$.

În particular, putem defini acum o semantică pentru expresia *(TRECUT P)* în raport cu un model $\mathbf{m}$ din structura de model Ω astfel:

$$m \vdash_\Omega (\text{TRECUT P}) \Longleftrightarrow \exists m' \in \Omega[m' < m, m' \vdash_\Omega P].$$

Altfel spus, *(TRECUT P)* este adevărat într-un model m dacă şi numai dacă există un alt model în structură care descrie un timp anterior lui m, în care P este adevărată.

Acest tip de analiză este numit *semantica universurilor posibile.*

11.2 Un model teoretic de semantică

Să considerăm o submulţime a *FOPC* formată numai din propoziţii şi conectori logici (fără termeni, variabile sau cuantificatori - a se vedea Anexa 1); ea este numită uzual *calcul propoziţional.* Un model pentru calculul propoziţional va asigna fiecărei formule (propoziţionale) una din valorile T (adevăr) sau F (fals). Construcţia este făcută definind aceste aplicaţii pentru propoziţiile simple (atomice) şi apoi realizând pe baza unor formule de calcul, valorile de adevăr pentru propoziţii compuse (construite cu ajutorul operatorilor logici). Ipoteza de bază folosită este aceea că înţelesul operatorilor logici poate fi definit numai în funcţie de valorile de adevăr ale argumentelor, independent de valoarea noii propoziţii.

Cei mai cunoscuţi operatori logici folosiţi în calculul propoziţional sunt:

Tabelul 11.1:

p	q	$\neg p$	$p\&q$	$p \vee q$	$p \supset q$	$p \equiv q$	$\neg p \vee q$
T	T	F	T	T	T	T	T
T	F	F	F	T	F	F	F
F	T	T	F	T	T	F	T
F	F	T	F	F	T	T	T

Formal, modelul semantic bazat pe calculul propoziţional este definit folosind o funcţie V numită *funcţie de valuare*, care asociază fiecărei propoziţii o valoare de adevăr din mulţimea $\{T, F\}$.

Astfel, fiind dată o logică conţinând o singură propoziţie P, se pot defini pentru ea două modele, prin două funcţii de valuare V_1, V_2 cu $V_1(P) = T, V_2(P) = F$.

În general, pentru o logică cu n propoziţii se pot defini 2^n funcţii de valuare posibile, deci 2^n modele pentru logica respectivă.

Orice funcţie de valuare V_m are următoarele proprietăţi deduse din Tabelul 11.1, care conduc la un calcul recursiv al funcţiei pentru propoziţii compuse, plecând de la propoziţii atomice:

Pentru orice propoziţii α, β:

1. $V_m(\alpha\&\beta) = V_m(\alpha)\&V_m(\beta)$
2. $V_m(\neg\alpha) = \neg V_m(\alpha)$
3. $V_m(\alpha \vee \beta) = V_m(\alpha) \vee V_m(\beta)$

4. $V_m(\alpha \supset \beta) = V_m(\alpha) \supset V_m(\beta)$

5. $V_m(\alpha \equiv \beta) = V_m(\alpha) \equiv V_m(\beta)$

Semnul egal (=) s-a pus între două expresii care pentru acelaşi set de valori asignat variabilelor, conduc la acelaşi rezultat.

În acest fel, fiind dat un model (adică o funcţie de valuare) se poate determina valoarea de adevăr a oricărei formule în calculul propoziţional căreia îi aparţine formula respectivă.

11.2.1 Logica folosită ca un model de gândire

Un prim **motiv de folosire** a *FOPC* este acela că el asigură o formulare precisă a noţiunilor de inferenţă şi argument valid (pentru precizări facem din nou trimitere la Anexa 1).

Exemplul 11.7 *Să plecăm de la următorul exemplu de argumentaţie:*

Toţi câinii iubesc copiii.
Grivei este un câine.
Deci Grivei iubeşte copiii.

Acest raţionament poate fi formalizat în FOPC printr-o formă de inferenţă, (regula modus-ponens) astfel:

$\forall x.D(x) \supset LB(x)$	*S-a notat cu D un predicat adevărat numai pentru câini,*
$\underline{D(GRIVEI1)}$	*iar cu LB un predicat adevărat numai pentru obiectele*
$LB(GRIVEI1)$	*care iubesc copiii.*

În practică, regulile de inferenţă nu sunt specifice unui anumit predicat; ele sunt exprimate folosind scheme în care propoziţiile sunt notate cu p, q, r, s, termenii cu a, b, c, d iar variabilele logice cu x, y, z. În acest mod se poate rescrie în general orice regulă de inferenţă.

Exemplul 11.8

$$\frac{p \supset q \qquad p}{q}$$
Modus ponens

$$\frac{\forall x.px}{pa}$$
Regula universal-particular

Mai multe reguli de inferenţă pot fi combinate pentru a trage concluzii; această manieră de lucru se numeşte *demonstraţie.* O demonstraţie constă dintr-un set de premise şi o secvenţă de formule, fiecare din ele obţinută din formulele precedente şi din premise, folosind o anumită regulă de inferenţă.

În general, pentru orice operator logic sunt definite apriori două reguli de inferenţă specifice: *regula de eliminare* şi *regula de introducere.*

Regula de introducere pleacă de la una sau mai multe premise care nu conţin operatorul şi ajunge la o concluzie în care operatorul este prezent.

Regula de eliminare ia ca premisă o formulă care conţine operatorul şi ajunge la o concluzie în care acest operator nu mai apare.

Tabelul 11.2:

Regula de introducere &

$$\frac{p \quad q}{p\&q}$$

Regula de eliminare &

$$\frac{p\&q}{p} \qquad \frac{p\&q}{q}$$

Regula de introducere ∨

$$\frac{p}{p \vee q}$$

Regula de introducere ∃

$$\frac{pa}{\exists x.Px}$$

Regula de introducere ¬

$$\frac{p \quad \text{verifică } q\&\neg q}{\neg p}$$

Regula de eliminare ¬

$$\frac{\neg\neg p}{p}$$

Ultima regulă (de introducere a negaţiei) necesită câteva precizări. Ea foloseşte o contradicţie; aceasta apare dacă există o formulă P pentru care se demonstrează că $P\&\neg P$ este adevărată. Intuitiv, demonstraţia foloseşte reducerea la absurd: pentru a arăta $\neg p$, se presupune p adevărată şi se arată că se ajunge la o contradicţie.

Fiind dat un set de reguli de inferenţă, există formule care se pot verifica fără nici o premisă. Acestea se numesc *tautologii* şi reflectă proprietăţi intriseci ale logicii.

Exemplul 11.9
$p \vee \neg p$ *(legea terţiului exclus)*
$\neg(p\&q) \equiv \neg p \vee \neg q$
$\neg(p \vee q) \equiv \neg p\&\neg q$ *(legile lui De Morgan)*

11.2.2 Relaţia dintre logică şi semantică

Pentru a putea fi folosită, o regulă de inferenţă trebuie să fie în primul rând *general valabilă*, adică să fie adevărată în toate modelele posibile.

Exemplul 11.10 *Să considerăm regula modus ponens (care poate fi privită şi ca o regulă de eliminare a implicaţiei):*

$$\frac{p \supset q \quad p}{q}$$

Parametrii p şi q iau valori din mulţimea propoziţiilor; vom considera valorile lor de adevăr. Pentru a arăta că această regulă de inferenţă este general valabilă, trebuie verificat că în orice model în care premisele sunt adevărate, concluzia trebuie să fie de asemenea adevărată. Conform Tabelului 11.1, $p \supset q$, p sunt simultan adevărate

numai în modelul 1; *aici, q este de asemenea adevărată. Deci regula este general valabilă, cel puţin în logica cu două propoziţii.*

Se ştie că un set de formule este *consistent* dacă există un model care asignează fiecărei formule din mulţime valoarea T. Spunem că un astfel de model *satisface* setul de formule.

Exemplul 11.11 *Setul de formule* $\{\neg P,\ P \vee Q,\ Q\}$ *este consistent, deoarece modelul* 3 *din Tabelul 11.1 ataşează fiecărei formule valoarea* T. *În schimb se poate arăta că setul de formule* $\{P \supset (Q \& R),\ P,\ \neg Q\}$ *nu este consistent, deoarece nu există nici un model care să le asocieze simultan valoarea* T.

În aceşti termeni se poate defini noţiunea de *tautologie.*

Definiţia 11.5 *O formulă f este o tautologie dacă în orice model i se asignează valoarea* T.

Se poate arăta imediat că formulele date în Exemplul 11.9 verifică definiţia tautologiei.

Se poate da acum definiţia formală a două relaţii sugerate în paragrafele anterioare:

- *suport*:

  Definiţia 11.6 *S este suport pentru P* ($S \vdash P$) *dacă şi numai dacă P ia valoarea T în toate modelele care satisfac S.*

- *demonstrabil*:

  Definiţia 11.7 *P poate fi demonstrat din S* ($S \models P$) *dacă şi numai dacă există o demonstraţie a lui P având ca singură premisă pe S.*

Problema finală care se pune este următoarea:
Fiind dat un set de reguli de inferenţă, este el suficient pentru a demonstra toate formulele ?
În termenii de sus, aceasta revine la a spune:

$$\text{Dacă } S \vdash P \text{ atunci } S \models P.$$

Un astfel de sistem de reguli se numeşte *complet.*

11.2.3 O semantică pentru *FOPC*

Pentru a asocia o semantică la calculul predicatelor de ordinul I, va trebui să extindem noţiunea de *expresie*. Astfel, termenii în *FOPC* nu vor reprezenta valori de adevăr ci obiecte fizice, evenimente, locaţii etc. Toate vor fi considerate ca fiind elemente ale unui domeniu unic general, notat cu Σ.

În calculul propoziţional (care era un subset al *FOPC*), funcţia de valuare asocia simbolurilor propoziţionale valorile T, F. În *FOPC* propoziţiile au ele însele o structură, iar funcţia de valuare trebuie să definească interpretarea constantelor, funcţiilor şi numelor de predicate. În particular, funcţia de valuare aplică fiecare termen într-un anumit element din domeniu.

Pentru a urmări mai uşor datele, să considerăm domeniul $\Sigma = \{\alpha, \beta, \gamma\}$ şi să luăm un model în care funcţia de valuare V_1 este definită pentru trei termeni $A1, B1, C1$ astfel:

$$V_1(A1) = \alpha, \qquad V_1(B1) = \beta, \qquad V_1(C1) = \gamma.$$

Funcţia de valuare asociază de asemenea numelor de predicate diverse tipuri de mulţimi, în funcţie de numărul de argumente. Să considerăm în prima fază numai predicate unare (cum ar fi *ROŞU*) care merg într-o submulţime a lui Σ (adică acele obiecte care au proprietatea de a fi *roşii*).

De exemplu, să presupunem că modelul defineşte predicatul *ROŞU* prin mulţimea $\{\alpha, \beta\}$: $V_1(\text{ROŞU}) = \{\alpha, \beta\}$.

Valoarea de adevăr a propoziţiilor simple construite dintr-un nume de predicat unar şi un argument poate fi acum definită prin:

$$V_1(P(a)) = \begin{cases} T & \text{dacă } V_1(a) \in V_1(P), \\ F & \text{altfel.} \end{cases}$$

(s-a notat cu P un predicat unar iar cu a un termen arbitrar).

Tabelul 11.3:

Model	A1	B1	C1	ROŞU	ROŞU(A1)	ROŞU(B1)	ROŞU(C1)
1.	α	β	γ	$\{\alpha\}$	T	F	F
2.	α	α	β	$\{\alpha\}$	T	T	F
3.	α	α	γ	$\{\alpha\}$	T	F	F
4.	α	α	α	$\{\alpha,\gamma\}$	T	T	T
5.	α	α	β	$\{\alpha,\gamma\}$	T	T	F
6.	α	α	α	$\{\beta,\gamma\}$	F	F	F
7.	α	α	β	$\{\beta,\gamma\}$	F	F	T

Fiind date definiţia lui V_1 şi regula care stabileşte valoarea de adevăr a propoziţiilor construite cu nume de predicate unare, se observă uşor că:

$$V_1(P(A1)) = T, \qquad V_1(P(B1)) = F, \qquad V_1(P(C1)) = F$$

Ca şi în cazul propoziţional, se pot construi tabele de adevăr pentru toate modelele posibile. Tabelul 11.3 listează numai câteva modele pentru logica cu trei constante şi numele de predicat unar *ROŞU*; un tabel complet ar avea 216 linii.

În cazul predicatelor de aritate n, valorile funcţiei de valuare sunt luate într-o mulţime de vectori cu n componente. Regula de calcul este:

$$V(P(a_1, \ldots, a_n)) = \begin{cases} T & \text{dacă } (V(a_1), \ldots, V(a_n)) \in V(P), \\ F & \text{altfel.} \end{cases}$$

Exemplul 11.12 *Luând predicatul binar "Iubeşte", el poate fi definit în modelul* 1, *în ipoteza că "α iubeşte γ" şi "β iubeşte α": V_1(Iubeşte)= $\{\alpha\gamma, \beta\alpha\}$.*

Acum se pot defini valorile de adevăr ale diverselor formule în acest model:

V_1 *(Iubeşte(A1,B1))=F,*
V_1 *(Iubeşte(B1,A1))=T,*
V_1 *(Iubeşte(A1,C1))=T.*

Regulile de definire a operatorilor logici sunt identice cu cele de la calculul propoziţional. Deci se pot calcula valorile de adevăr pentru diverse formule mai complicate; de exemplu, conform asignărilor date mai sus:

$$\neg\ (Roşu(A1) \vee Iubeşte(A1,B1))=F.$$

11.3 Semantici pentru cuantificatori

Pentru a defini semantici ale formulelor care conţin variabile cuantificate, trebuie să existe posibilitatea de a face substituţii ale variabilelor în formule. Pentru aceasta, definim $V_{i\{x/\alpha\}}$ ca o extensie a funcţiei de valuare, care este similară funcţiei V_i, numai că defineşte evaluarea variabilei x în domeniul dat de constanta α.

Exemplul 11.13 *Pentru funcţia de valuare V_1 definită în secţiunea anterioară,*

$$V_{1\{x/\alpha\}}(x) = \alpha, \qquad V_{1\{x/\alpha\}}(Roşu) = V_1(Roşu) = \{\alpha\}.$$

Deci $V_{1\{x/\alpha\}}(Roşu(x)) = T$ deoarece $V_{1\{x/\alpha\}}(x) \in V_{1\{x/\alpha\}}(Roşu)$.

În aceste condiţii, o semantică a formulelor care conţin cuantificatori este definită de următoarele condiţii:

$V_m(\forall x.P) = T$ dacă pentru orice $\alpha \in \Sigma$, $V_{m\{x/\alpha\}}(P) = T$
$V_m(\exists x.P) = T$ dacă există cel puţin un $\alpha \in \Sigma$ astfel încât $V_{m\{x/\alpha\}}(P) = T$.

Toate definiţiile precedente de suport, demonstrabilitate, completitudine etc. pot fi considerate fără modificări şi în cazul semanticilor pentru *FOPC*. În particular, se poate arăta că un set de formule este consistent construind un model în care toate formulele iau valoarea adevăr.

Exemplul 11.14 *Să luăm setul de formule*

$$\{\forall x.Q(x) \supset R(x),\ Q(A1),\ Q(B1),\ \neg Q(C1)\}$$

Pentru a arăta că este consistent, definim un model astfel:
Fie $\Sigma = \{\alpha, \beta, \gamma\}$ unde $V(A1) = \alpha, V(B1) = \beta, V(C1) = \gamma$.
Să luăm $V(Q) = \{\alpha, \beta\}, V(R) = \Sigma$.
Se verifică uşor că în acest model, toate formulele din setul ales iau valoarea T.

Prelegerea 12

Reprezentarea universului cunoaşterii

12.1 Reprezentarea cunoştinţelor

Există două forme de cunoaştere, cruciale în orice sistem de reprezentare a cunoştinţelor: *cunoaşterea generală* a universului şi *cunoaşterea specifică* a situaţiei curente.

Anumite aspecte ale cunoaşterii universului au fost deja considerate şi introduse în ierarhia tipurilor, în relaţiile dintre ele, în structurile sintactice şi semantice ale limbii etc. Este vorba de informaţii privind restricţii generale ale universului definite în termenii limbajului. În majoritatea lor, elementele de cunoaştere generală sunt specificate în caracteristicile tipurilor de obiecte din univers; nu este necesară o specificare individuală a fiecărui element.

Această cunoaştere generală este esenţială pentru a rezolva probleme de interpretare din multe limbaje; una din ele este *ambiguitatea de exprimare*.

Exemplul 12.1 *Modul de ataşare a PP finale din următoarele două propoziţii depinde numai de nivelul de cunoştinţe al cititorului în domeniul respectiv:*

Am urmărit evoluţia 10 *minute.*

Am urmărit evoluţia zece milioane de ani.

Cunoaşterea specifică este importantă în multe situaţii, cum ar fi de exemplu determinarea persoanei la care se referă construcţiile substantivale, sau eliminarea sensurilor ambigui ale cuvintelor bazate pe situaţia curentă.

Definiţia 12.1 *O reprezentare a cunoştinţelor constă dintr-o bază de date de propoziţii numită bază de cunoştinţe (KB) şi un set de tehnici de inferenţă folosite pentru a deduce noi propoziţii plecând de la KB curent.*

Un set de tehnici de inferenţă este **rezonant** *dacă generează numai propoziţii adevărate când toate propoziţiile din KB sunt adevărate.*

După cum vom vedea, această condiţie de rezonanţă nu este o condiţie necesară.

Limbajul în care sunt definite propoziţiile din KB este numit *limbajul de reprezentare a cunoştinţelor (KRL)*. Acesta poate fi acelaşi cu cel folosit la limbajul formei logice, dar este preferabilă diferenţierea lor.

Să considerăm de exemplu tratarea cuantificatorilor. În limbajul formei logice au fost definiţi un număr mare de cuantificatori, lucru motivat de numeroasele sensuri oferite de limbajul natural. În multe limbaje de reprezentare de cunoştinţe sunt definiţi uzual un număr mult mai mic - şi practic folosit numai unul: o construcţie asociată cuantificatorului universal.

De aceea se recomandă folosirea a două limbaje de reprezentare separate şi utilizarea unor funcţii pentru a face legătura între ele.

Când o formulă P trebuie să fie adevărată pe baza formulelor din KB sau a formulelor care reprezintă înţelesul unei propoziţii date, spunem că KB (respectiv propoziţia) *suportă* P. Multe concluzii necesare în înţelegerea limbajului nu sunt bazate pe suporturi, dar sunt totuşi *implicaţii* ale propoziţiei.

Definiţia 12.2 *Implicaţiile sunt concluzii tipice trase dintr-o propoziţie, care pot fi infirmate în anumite circumstanţe.*

Exemplul 12.2 *Propoziţia*

Matei are două maşini.

suportă faptul că

Matei are maşină.

(acesta nu poate fi negat) şi implică faptul că

Matei nu are trei maşini.

Această ultimă observaţie poate fi negată dacă discuţia continuă cu

De fapt el are trei.

Sistemele KR trebuie să accepte ambele forme de inferenţă.

12.1.1 Tipuri de inferenţă

Tehnicile de inferenţă se împart în *forme deductive* şi *nedeductive*. Formele deductive de inferenţă sunt justificate de notaţia logică a suportului. Fiind dat un set de fapte, un proces de inferenţă deductivă va genera concluzii care se desprind logic din acele fapte.

Inferenţa nedeductivă se împarte în mai multe clase. Astfel, avem deducerea elementelor generale plecând de la exemple (*inferenţa inductivă*), sau a cauzelor studiind efectele (*inferenţa abductivă*).

Inferenţa inductivă - de exemplu - reprezintă modul principal de lucru în prezentarea procedeelor lingvistice din prelucrarea limbajului natural.

Inferenţa abductivă lucrează de obicei invers faţă de cea deductivă; pentru exemplificare, să considerăm axioma

$$A \supset B$$

Inferenţa deductivă va folosi această axiomă pentru a defini B atunci când se dă A. Inferenţa abductivă este folosită pentru a construi A atunci când se dă B, deoarece A este o raţiune pentru ca B să fie adevărată.

Multe sisteme permit folosirea informaţiei incomplete.

Definiţia 12.3 *O regulă incompletă este o regulă de inferenţă care poate fi exceptată.*

Să notăm folosirea informaţiei incomplete prin $\Longrightarrow$. Atunci o regulă de inferenţă incompletă va lucra astfel:

Dacă $A \Longrightarrow B$, *A este adevărată şi* $\neg B$ *nu este demonstrabilă, atunci se poate deduce B.*

Exemplul 12.3 *Propoziţia*

Păsările zboară.

poate fi reprezentată în FOPC prin

$\forall x PAS\breve{A}RE(x) \Longrightarrow ZBOAR\breve{A}(x)$

Aceasta are ca efect faptul că ori de câte ori există o pasăre B pentru care nu este demonstrabil că $\neg$ *ZBOARĂ(B), atunci se poate presupune că ZBOARĂ(B). Cu alte cuvinte, despre o anumită pasăre se poate spune că este capabilă să zboare numai dacă nu s-a afirmat explicit că nu poate face acest lucru. De exemplu, struţul este menţionat separat ca o pasăre care nu zboară.*

Regulile incomplete introduc un nou set de complexităţi în reprezentare. Fără astfel de reguli, majoritatea reprezentărilor sunt *monotone*, deoarece prin adăugarea de noi afirmaţii creşte doar numărul de formule suportate. Formal,

Definiţia 12.4 *O reprezentare este monotonă dacă verifică următoarea condiţie: Dacă baza sa de cunoştinţe - KB - suportă o concluzie C şi la KB se adaugă o formulă nouă construind o bază consistentă KB', atunci KB' va suporta de asemenea C.*

Acest lucru nu este adevărat într-o reprezentare care foloseşte reguli incomplete. Astfel de reprezentări se numesc *nemonotone*.

Exemplul 12.4 *Să considerăm baza de cunoştinţe K alcătuită din formulele:*

Pisică (Puffy)	*Puffy este o pisică.*
Siamez(Pisică (Puffy))	*Puffy este o pisică siameză.*
$\forall c.Pisic\breve{a}(c) \Longrightarrow Toarce(c)$	*Pisicile torc.*

De aici, folosind informaţia incompletă se poate deduce "Toarce(Puffy)", pentru că nu există nici o informaţie care să contrazică noua afirmaţie. Dar dacă se adaugă o nouă formulă care spune că nici o pisică siameză nu toarce, atunci noua bază de cunoştinţe nu va mai suporta faptul că "Puffy toarce".

Concluziile nemonotone nu sunt specifice informaţiilor incomplete; există şi alte tehnici de introducere a lor. Astfel, în ipoteza universului minim (*CWA* - **C***losest* **W***orld* **A***ssumption*) se poate reţine informaţie incompletă despre anumite predicate. Atunci,

despre un predicat P pentru care CWA este valabilă, dacă o propoziţie folosind P nu poate fi demonstrată într-o KB, atunci negaţia ei este presupusă corectă.

Exemplul 12.5 *Să considerăm o bază de date pentru programul zborurilor unei companii de avioane.*

KB stochează informaţia care există despre zboruri - de exemplu:

$FDG - 100$ *zboară de la Roma la Paris.*

dar nu conţine explicit nici o informaţie negativă - cum ar fi

$FDG - 100$ *nu zboară la Londra.*

sau

Nu există nici un zbor $FDG - 231$.

Astfel de informaţii pot fi doar deduse dacă procesul de inferenţă lucrează în ipoteza universului minim despre zboruri.

Se poate dezvolta un model de semantică teoretică pentru logici nemonotone folosind conceptul de *model minimal.* Să considerăm de exemplu ipoteza universului minimal pentru un predicat P. Se poate defini atunci o ordonare totală pe modelele lui KB astfel:

$$m_1 <_P m_2 \Longleftrightarrow I_{m_1}(P) \subseteq I_{m_2}(P)$$

Deci, un model m_1 este mai mic decât un model m_2 în raport cu un predicat P dacă şi numai dacă mulţimea obiectelor x pentru care $P(x)$ este adevărată în m_1 este o submulţime a mulţimii obiectelor x cu $P(x)$ adevărată în m_2.

Odată definită această ordonare, modelul minimal în raport cu P constă din mulţimea

$$\{\mathrm{m} \mid \mathrm{m}\ \mathit{model} \quad \neg\, \exists\, \mathrm{m}' <_P \mathrm{m}\ \}.$$

O altă modalitate de a formaliza ipoteza universului minim este de a adăuga o axiomă la KB care să închidă suportul. Ea este numită *axioma de completare.*

Exemplul 12.6 *Să considerăm un KB conţinând propoziţiile*
$\{P(A),\ P(B),\ Q(C),\ Q(A)\ \}$.
Atunci axioma de completare pentru P este

$$\forall x.P(x)\ \equiv\ (x = A \vee x = B).$$

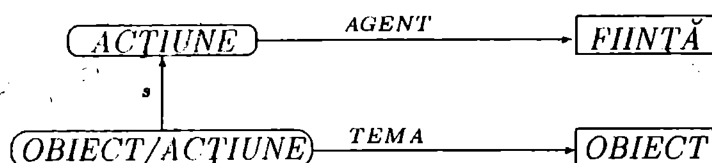
Ea spune că P este adevărată numai pentru valorile A sau B ale argumentului. Se poate arăta că mulţimea modelelor, extinsă cu aceasta axiomă axiomă de completare, coincide cu mulţimea modelelor minimale (în raport cu P) a KB iniţial.

12.1.2 Tehnici de inferenţă

În sistemele de reprezentare a cunoştinţelor există două clase mari de tehnici de inferenţă: *procedurale* şi *declarative.* Majoritatea sistemelor combină aceste tehnici acoperind o paletă largă de variante, de la sistemele pur declarative la cele pur procedurale.

Primele sunt bazate pe demonstrarea automată. KB este reprezentat printr-un sistem de axiome, iar inferenţa este realizată folosind un algoritm deductiv de demonstrare automată a teoremelor. Accentul cade aici pe asignarea unei semantici formale expresiilor, independent de componenta "inferenţă".

Figura 12.1:



Sistemele pur procedurale pe de altă parte folosesc aspectele inferenţiale ale reprezentării; expresiile într-o KB pot să nu aibă nici un înţeles independent de manipularea lor în program.
Ca un exemplu putem da un sistem care foloseşte procedurile aritmetice proprii calculatorului pentru a evalua expresiile aritmetice, fără nici o reprezentare explicită a cunoştinţelor despre matematică (de pildă, axiomele lui Peano).
Adesea eficiente, aceste tehnici sunt greu de analizat din cauza lipsei formalismului specific aplicaţiei.

Exemplul 12.7 *În prelegerile anterioare s-a introdus noţiunea de "reţele semantice". Procesul lor de inferenţă poate fi realizat procedural sau declarativ. O modalitate pur declarativă va modela fiecare fapt despre subtipuri şi roluri ca o axiomă, iar proprietăţile de dependenţă vor rezulta din inferenţa standard deductivă. Astfel, pentru reţeaua din Figura 12.1, informaţia poate fi reprezentată prin axiomele (scrise în FOPC):*

1. $\forall x.ACŢIUNE(x) \supset \exists a.AGENT(x,a) \& FIINŢĂ(a)$
2. $\forall a \exists x.ACŢIUNE(x) \& AGENT(x,a) \supset FIINŢĂ(a)$
3. $\forall x.OBIECT/ACŢIUNE(x) \supset ACŢIUNE(x)$
4. $\forall x.OBIECT/ACŢIUNE(x) \supset \exists o.TEMA(x,o) \& OBIECT(o)$
5. $\forall o \exists x.OBIECT/ACŢIUNE(x) \& TEMA(x,o) \supset OBIECT(o)$

Folosind aceste axiome, se poate demonstra că rolul AGENT este moştenit de clasa OBIECT/ACŢIUNE. Altfel spus, pentru orice obiect A astfel încât OBIECT/ACŢIUNE(A) este adevărat, se poate demonstra că A are rolul agent; deci, folosind axiomele 3 şi 1,

$$\exists a.AGENT(A,a) \& FIINŢĂ(a).$$

O versiune procedurală a acestei construcţii va fi un program care pleacă de la un nod OBIECT/ACŢIUNE precizat, află toate rolurile ataşate lui, apoi traversează arcul s până la supertipul ACŢIUNE şi află toate rolurile ataşate aici. Răspunsul este dat de mulţimea totală de roluri colectate de această procedură. Deci, în particular, orice OBIECT/ACŢIUNE are un rol AGENT moştenit de la clasa ACŢIUNE.

12.2 O reprezentare bazată pe *FOPC*

Plecând de la *FOPC* se defineşte un *KRL* care este o extensie pentru a cuprinde şi informaţiile specifice reprezentării cunoştinţelor.

Termenii limbajului sunt constante (exemplu *Ion*), funcţii (exemplu *tată (Ion)*) şi variabile $(x, y, \ldots)$. De remarcat - ca o diferenţă - că limbajul formei logice nu avea definite constante; aici (în *KB*) constantele sunt folosite pentru a reprezenta singularităţi specifice.

De exemplu, termenul din forma logică *(NUME j1 "Ion")* reprezintă înţelesul unei afirmaţii referitoare la *Ion*. Persoana prezentă la care se face referire într-un context dat poate fi reprezentată în *KB* de constanta *Ion*.

Similar notaţiei domeniului cuantificatorilor generalizaţi din limbajul formei logice, se procedează şi în *KRL*: restricţia domeniului se notează prin variabila cuantificată, urmată de " :".

Exemplul 12.8

$\exists x : Om(x)\ Fericit(x)$	*este echivalent cu*	$\exists x.Om(x) \& Fericit(x)$
$\forall x : Om(x)\ Fericit(x)$	*este echivalent cu*	$\forall x.Om(x) \supset Fericit(x)$

Aici este necesar şi predicatul egalitate $(a = b)$, care semnifică faptul că termenii a şi b au aceeaşi notaţie.

> ***Fiind dată o propoziţie simplă P_a care foloseşte constanta a, dacă P_a este adevărată şi $a = b$, atunci P_b trebuie să fie adevărată.***

Multe sisteme de reprezentare a cunoştinţelor nu folosesc explicit cuantificatori. Ele pot introduce totuşi variabile, care acţionează ca variabile cuantificate universal cu domeniul vid.

Exemplul 12.9 *Formula*

$$(P\ ?x\ A)$$

scrisă în KB corespunde ca înţeles formulei din FOPC

$$\forall x.P(x, A).$$

Variabilele cuantificate existenţial sunt prelucrate cu o tehnică numită *skolemizare* care înlocuieşte variabila cu o constantă nouă, nedefinită anterior.

Exemplul 12.10 *Formula*

$$\exists y \forall x.P(x, y)$$

va fi codificată în KB în formula

$$(P\ ?x\ Sk1)$$

unde Sk1 este o constantă nouă, neutilizată anterior, fixată pentru obiectul despre care se ştie că există.

Dependenţele de domeniu ale cuantificatorilor sunt indicate folosind funcţii noi, numite *funcţii Skolem*. De exemplu, formula

$$\forall y \exists x.P(x, y)$$

va fi codificată în KB în formula

$$(P\ (Sk2\ ?y)\ ?y)$$

unde $Sk2$ este o funcţie nouă, care produce un obiect (eventual) distinct pentru fiecare valoare a lui $?y$.

Formulele vor fi reprezentate adesea în formate care combină ambele modalităţi de scriere: cuantificatorii universali sunt scrişi explicit, iar variabilele existenţiale sunt skolemizate.

Exemplul 12.11 *Formula*

$$\forall y \exists x.P(x, y)$$

poate fi scrisă şi

$$\forall y\ P\ (Sk1(y),\ y).$$

De fapt, după cum se vede, toate aceste forme de scriere sunt echivalente.

Stilul prezentării în definirea limbajului nu are nici o restricţie (în afară de aceea de a fi conformă cu notaţia *FOPC*). În particular, nu se poate spune nimic despre forma predicatelor. Există o plajă exprem de largă de selecţie. La o extremă se pot defini predicate diferite pentru fiecare sens al cuvântului (strategia fundamentală folosită în limbajul formei logice). La cealaltă extremă, se poate folosi un set predefinit de predicate numite *primitive*, şi sensurile cuvintelor vor fi scrise în termenii acestor primitive.

Fiecare modalitate are avantajele şi dezavantajele sale. De obicei ele sunt combinate folosind tehnica ierarhiei tipurilor.

De multe ori, o reprezentare a cunoştinţelor trebuie să extragă informaţie din modalităţi diverse de definire a sensurilor cuvintelor.

Uneori se ştie o definire completă a unui termen. De exemplu, se poate defini predicatul *tată* ca părinte mascul, adică

$$\forall\ x.\text{TATĂ}(x) \equiv \exists\ y\ \text{PĂRINTE}(x, y) \& MASCUL(x)$$

Majoritatea cuvintelor nu sunt însă definite aşa de precis. Astfel, nu există nici un set de proprietăţi care să definească precis noţiuni uzuale cum ar fi ***câine, pisică, scaun*** etc. Ele pot fi clasificate în ierarhii de tipuri, fiecare însoţită de axiomele aferente; dar nici o definiţie precisă nu poate fi dată. Privite ca axiome în *FOPC*, ele vor folosi doar implicaţii.

Exemplul 12.12 *O axiomă pentru CÂINE1 poate fi*

$$\forall\ x.CÂINE(x) \supset CANIN(x) \& ANIMAL_DOMESTIC(x)$$

unde CANIN va fi definit la rândul lui ca un tip de MAMIFER ş.a.m.d.

Astfel de axiome stabilesc multe proprietăţi esenţiale pe care trebuie să le aibă un obiect ca să fie recunoscut drept "câine", dar conceptul nu se poate defini complet. De exemplu, cineva poate creşte un pui de lup, care are toate proprietăţile unui câine, dar nu este câine.

Din punct de vedere al propoziţiilor generate, majoritatea predicatelor din limbajul de reprezentare sunt abstractizări ale cuvintelor limbajului natural. Dificultatea apare la generarea de propoziţii bazate pe un anumit înţeles.

De exemplu, să presupunem că avem formula:

$\forall p.((Mascul_Om\ p)\&\exists c.Părinte(p,c)).$
$Merge_Cu_Maşina(p, L1)\&Clădire(L1)\&\ Destinată_învăţării(L1)$

care are o realizare naturală prin propoziţia

Toţii taţii şofează spre şcoală.

Pentru a genera o astfel de afirmaţie, sistemul trebuie să fie capabil să interpreteze formula

$$\forall\ p.((Mascul_Om\ p)\&\ \exists c.Părinte(p,c))$$

sub forma

fiinţă umană de genul masculin care are un copil,

să considere propoziţia

$Merge_Cu_Maşina\ (p, L1)\&\ Clădire\ (L1)\&\ Destinată_învăţării\ (L1)$

care literar se poate trata drept

se deplasează cu maşina spre o clădire folosită pentru învăţătură,

şi apoi să le comprime în propoziţia dată anterior.

Evident, aceasta înseamnă un volum substanţial de cunoştinţe despre înţelesul unor cuvinte specifice cum ar fi *tată* sau *şofează* şi un proces complicat de combinare a formulelor în *KRL*. Dacă nu există predicate corespunzătoare acestor semnificaţii în *KRL*, procesul devine foarte complicat. Dacă însă există predicate pentru cuvintele solicitate, organizarea lor într-o manieră ierarhică permite diverse metode pentru identificarea realizărilor posibile ale unei formule.

Revenind la exemplu, fiind dat predicatul abstract *Mascul_Om*, se pot cerceta toate predicatele aflate pe nivelele inferioare într-o astfel de ierarhie de abstractizare, pentru a găsi unul care să reflecte concis înţelesul dorit. Predicatul *tată* va fi o alegere bună pentru că suportă nu numai *Mascul_Om* dar şi a doua parte a înţelesului, anume $\exists c.Părinte(p,c)$.

Ideea folosită în desemnarea unei reprezentări efective a cunoştinţelor este de a alege setul de predicate cu o structură ierarhică cât mai bogată.

12.3 Cadrul ca reprezentare a informaţiei

Mare parte din inferenţa cerută de înţelegerea limbajului natural foloseşte prezumţii despre ceea ce este tipic la anumite obiecte sau situaţii aflate în discuţie. Astfel de informaţie este codificată în structuri numite *cadre*.

Definiţia 12.5 *Un cadru este o colecţie de fapte şi obiecte care descriu un anumit obiect (sau situaţie), împreună cu strategiile de inferenţă specifice noilor situaţii care pot apare.*

Situaţia reprezentată poate varia de la scene vizuale la structura obiectelor fizice complexe. Sistemele bazate pe cadre oferă facilităţi cum ar fi raţionamentul cu reguli incomplete, descendenţă automată a proprietăţilor pe baza ierarhiilor sau raţionamente procedurale.

Obiectele principale într-un cadru sunt numele asignate, numite *sloturi* sau *roluri* (similar rolurilor tematice de la forma logică). De exemplu, cadrul pentru *casă* poate avea drept sloturi *bucătăria, sufrageria, holurile, uşa de la intrare* etc.

Cadrul specifică de asemenea şi relaţiile între sloturi şi obiectele reprezentate în cadru. De exemplu, slotul *bucătărie* din cadrul *casă* trebuie încadrat fizic în interiorul casei şi să conţină diverse unelte necesare preparării mâncării.

Fiecare slot poate fi considerat o funcţie care ia un obiect descris de cadru (o instanţă a cadrului) şi produce valoarea sa corespunzătoare.

Deci o instanţă particulară a cadrului *casă* - să o notăm cu *H1* - constă dintr-o instanţă particulară a unei *bucătării*, la care se poate face referire sub forma *bucătărie(H1)*, plus alte instanţe particulare ale tuturor celorlaltor sloturi.

Exemplul 12.13 *Definiţia cadrului unui calculator personal poate fi scrisă:*

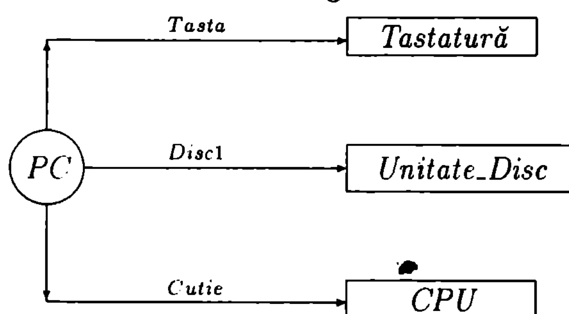
Clasa de obiecte: **PC(e)**

Roluri: *Tasta, Disc1, Cutie*

Restricţii: *Tastatură(Tasta), Unitate_Disc(Disc1), CPU(Cutie)*

Această structură reprezintă ideea că toate obiectele de tipul PC au sloturi de tip tastatură, unitate_disc şi Unitate centrală (CPU) - identificate respectiv prin funcţiile Tasta, Disc1 şi Cutie. Se poate da şi o reprezentare grafică (Figura 12.2):

Figura 12.2:



O apariţie de tipul PC - să spunem PC3 - având ca subcomponente Tast13, Dc11 şi CPU00023 va fi reprezentată în notaţia cadru prin

$$(PC3 \text{ este } PC \text{ cu } Tasta = Tast13,\ Disc1 = Dc11,\ Cutie = CPU00023)$$

Această definiţie poate fi privită ca o prescurtare a formulei FOPC:

$$PC(PC3)\ \&\ Tasta(PC3) = Tast13\ \&\ Disc1(PC3) = Dc11\ \&$$
$$Cutie(PC3) = CPU00023$$

12.3.1 Sloturi cu restricţii

Reprezentarea cadrelor poate fi folosită pentru a codifica informaţie suplimentară despre subcomponente. Un exemplu des întâlnit în înţelegerea limbajului natural constă în reprezentarea acţiunilor. Cunoaşterea situaţiilor obişnuite în care apare o acţiune poate fi foarte utilă în interpretarea limbajului. Este vorba mai ales de cunoaşterea relaţiei cauză - efect şi cunoaşterea condiţiilor tipice necesare apariţiei acţiunii. Notaţia slotului este extinsă pentru a reţine relaţiile între apariţia cadrului şi alte propoziţii sau evenimente.

Pentru acţiuni, sunt utile următoarele relaţii:

precondiţii	–	proprietăţi care declanşează de obicei acţiunea;
efecte	–	proprietăţi cauzate de obicei de acţiune;
descompunere	–	modul în care este realizată de obicei acţiunea (definită ca o secvenţă de subacţiuni).

Exemplul 12.14 *Figura 12.3 arată definiţia acţiunii de a cumpăra ceva. Acţiunea foloseşte 4 obiecte: cumpărătorul, vânzătorul, obiectul şi o sumă de bani egală cu preţul obiectului.*

Definiţia arată că acţiunea se poate declanşa numai dacă cumpărătorul are destui bani şi vânzătorul are obiectul (precondiţiile); tipic, la sfârşit cumpărătorul deţine obiectul iar vânzătorul are banii (efectele). În sfârşit, un mod tipic de acţiune este acela în care cumpărătorul dă banii vânzătorului iar acesta dă obiectul cumpărătorului (descompunere).

Figura 12.3:

		Clasa de acţiune: CUMPĂRĂ(b)
Roluri	:	*Cumpărător, Vânzător, Obiect, Bani*
Restricţii	:	*Om(Cumpărător),*
		Agent_Vânzări(Vânzător),
		EObiect(Obiect),
		Valoare(Bani),
		Preţ(Obiect)
Precondiţii		*DEŢINE(Cumpărător, Bani),*
		DEŢINE(Vânzător, Obiect)
Efect	:	$\neg$ *DEŢINE(Cumpărător, Bani),*
		$\neg$ *DEŢINE(Vânzător, Obiect),*
		DEŢINE(Cumpărător, Obiect),
		DEŢINE(Vânzător, Bani)
Descompunere	:	*DĂ(Cumpărător, Vânzător, Bani),*
		DĂ(Vânzător,Cumpărător, Obiect)

12.4 Folosirea mulţimilor în definirea cuantificatorilor

Având definit un *KRL*, să facem câteva observaţii referitoare la aplicarea limbajului formei logice în *KRL*. Cele mai mari diferenţe au apărut în scrierea cuantificatorilor (foarte numeroşi în limbajul formei logice, maxim doi în *KRL*). Această diferenţă de număr poate fi redusă extinzând definiţia *KRL* şi asupra obiectelor.

KRL poate lucra cu mulţimi, cu condiţia ca acestea să fie finite. O mulţime poate fi indicată listând elementele ei (de exemplu $\{Ana1, Cici1\}$).

Mulţimile sunt notate în mod uzual cu constante; astfel, S_1 poate fi o mulţime definită prin formula $S_1 = \{Ana1, Cici1\}$.

Proprietăţile din teoria mulţimilor sunt valabile şi în *KRL*. Operaţii cum ar fi *reuniunea, intersecţia, incluziunea, apartenenţa* etc. apar sub denumirea de *predicate.*

Folosirea mulţimilor în definiţii se face obişnuit; de exemplu, propoziţia

Unii s-au întâlnit la trei.

produce interpetarea:

$$\exists M : M \subset \{x | OM(x)\}.\text{ÎNTÂLNI}(M, 3PM)$$

Prin convenţie, numele mulţimilor se vor scrie cu majuscule. Sunt anumite verbe care în unele argumente operează numai cu mulţimi. Altele dimpotrivă.

Să considerăm câteva formule care apar în urma citirilor colective/distributive. Astfel, există două interpretări ale propoziţiei

Câţiva oameni au cumpărat un costum.

care are următoarea formă logică (s-a omis operatorul de timp):

(CÂŢIVA **m1**:*(PLUR OM1)*
(UN **s1**:*(COSTUM1*
(CUMPĂRA1 **m1 s1**)))

Citirea colectivă va aplica această formă în

$$\exists M1 : M1 \subset \{z | OM(z)\}\ \exists s : COSTUM(s).CUMPĂRA1(M1, s)$$

Citirea distributivă conduce la formula:

$$\exists M2 : M2 \subset \{z | OM(z)\}\ \forall m : m \in M2\ \exists s : COSTUM(s).CUMPĂRA1(m, s)$$

Se observă că diferenţa principală dintre cele două scrieri constă în tipul de argument folosit de predicatul *CUMPĂRA1*: o mulţime ($M1$) la citirea colectivă, un element (m) la cea distributivă.

Reprezentarea folosind mulţimi poate fi folosită pentru informaţii suplimentare; astfel putem arăta că mai mulţi oameni (nu numai unul) cumpără un costum. Această informaţie se poate codifica definind o funcţie care dă cardinalul $|S|$ al unei mulţimi S.

Exemplul 12.15 *Propoziţia*

Trei oameni au intrat în cameră.

are reprezentarea (s-a omis operatorul timp):

$$\exists M : (M \subset \{y|OM(y)\} \ \& \ |M| = 3) \ \forall m : m \in M.INTRA1(m, CAMERA1)$$

Schimbând egalitatea $|M| = 3$ *cu* $|M| \geq 3$, *se obţine înţelesul propoziţiei*

În cameră au intrat cel puţin trei oameni.

Folosind mulţimi se pot aproxima şi alţi cuantificatori din limbajul formei logice. Astfel, dacă cuantificatorul *mulţi* se poate defini prin *cel puţin jumătate*, atunci

Mulţi oameni râdeau.

are înţelesul

$$\exists M : (M \subset \{y|OM(y)\} \ \& \ |M| \leq |\{y|Om(y)\}|/2) \ \forall m : m \in M.RÂDE1(m)$$

Prelegerea 13

Raţionament şi cunoaştere

13.1 Verbele în reprezentarea cunoaşterii

Una din componentele centrale în orice reprezentare a cunoaşterii constă în tratarea verbelor şi a timpului. Multe limbaje folosesc timpul incluzând informaţia temporală implicit în timpul verbelor şi explicit în adverbe temporale (de exemplu *pentru cinci minute, la ora* 3, *după ce am plecat* etc.).

În limbajul formei logice, informaţia temporală era manevrată în mai multe moduri. Astfel, existau operatorii modali pentru reprezentarea timpului (*TRECUT, PREZ* etc), conectori temporali (*ÎNAINTE, ÎN_TIMPUL*, ...), iar toate predicatele puteau avea argumente de timp.

Există mai multe tipuri diferite de timp. Din acest punct de vedere, avem *punct temporal, interval temporal, durată.* Pentru ele se definesc următoarele relaţii:

$t_1 < t_2$	–	punctul/intervalul t_1 este înaintea punctului/intervalului t_2;
$t_1 : t_2$	–	intervalul t_1 continuă cu intervalul t_2, *sau*
	–	punctul t_1 defineşte începutul intervalului t_2, *sau*
	–	punctul t_2 defineşte sfârşitul intervalului t_1;
$t_1 \subseteq t_2$	–	punctul/intervalul t_1 este conţinut în punctul/intervalul t_2.

Propoziţiile descriu afirmaţii care aparţin unei anumite clase: cele care descriu stări (afirmaţii *statice*), cele care descriu activităţi de deplasare (afirmaţii *active*) şi cele care descriu evenimente complete (afirmaţii *totale*).

Definiţia 13.1 *Afirmaţiile statice descriu anumite proprietăţi ale universului, care pot avea loc (pentru un moment sau pentru o perioadă nedefinită).*

De exemplu:

Mihai este fericit.
Eu cred că pământul este plat.

De asemenea, ele descriu situaţii care nu au un punct final precis şi nu pot apare în anumite contexte lingvistice:

* *Mihai fiind fericit.*
* *Eu crezând că pământul este plat.*

Definiţia 13.2 *Afirmaţiile active descriu activităţi care pot apare într-un interval de timp.*

De exemplu:

Mihai aleargă.
Uşa se deschide şi se închide permanent.

Propoziţiile care descriu stări sau activităţi nu folosesc de obicei operatori temporali (cum ar fi *cinci minute*), dar au modificatori de durată (*pentru cinci minute*).

Definiţia 13.3 *Afirmaţiile totale descriu acţiuni finalizate.*

Astfel, următoarele afirmaţii sunt totale:

Paul a reuşit să adoarmă.
Silvia s-a urcat pe munte.

În ambele propoziţii, evenimentul se termină la un anumit moment (numit *punct culminant*) şi rezultă o anumită proprietate care începe din acest punct. În exemplele de sus, ştim că Paul este adormit la sfârşitul evenimentului, iar în a doua, că Silvia se află pe vârful muntelui.

Propoziţiile care includ afirmaţii totale pot folosi modificatori temporali; de exemplu:

Au suit muntele în două zile.
Paul a reuşit să adoarmă într-o oră.

Eventualităţile totale pot fi clasificate în două subclase, după felul cum descriu numai o tranziţie (clasa *finalizare*) sau prezintă o activitate care duce la punctul culminant (clasa *completare*). Exemplele de sus descriu finalizări; pentru completări putem exemplifica prin:

Sorin a recunoscut persoana.
Elena l-a lămurit.

Cele patru tipuri de clase de afirmaţii pot fi distinse prin tipuri diferite de argumente temporale. Astfel:

- Afirmaţiile statice pot fi adevărate într-un punct sau un interval; în acest din urmă caz, spunem că sunt *omogene*. Evident, dacă o afirmaţie este omogenă pe un interval, ea va fi omogenă pe orice subinterval al său.

- Propoziţiile de finalizare, cum ar fi

  Paul a atins vârful.

  Rodica a închis uşa.

  corespund propoziţiilor care descriu tranziţii. Astfel, în exemplele date, prima descrie o tranziţie după care Paul se află în vârf, în timp ce a doua descrie o tranziţie după care uşa este închisă. Asemenea predicate nu pot fi adevărate pe intervale dar pot oferi informaţie relativ la poziţia statică rezultată.

Exemplul 13.1

$$\forall\, a1, l1, t1\ .ATINGE(a1, l1, t1) \supset \forall\, T1.\, t1 : T1\ \&\ LA(a1, l1, T1)$$

se interpretează astfel: dacă un agent $a1$ *atinge o locaţie* $l1$ *la momentul* $t1$*, atunci* $a1$ *stă în* $l1$ *un anumit interval de timp care începe din* $t1$*.*

- Propoziţiile care descriu procese corespund verbelor active (*Ion alerga*). Predicatele unor astfel de procese apar numai pe intervale şi tind să fie omogene, deşi nu într-un mod atât de strict ca cele statice. În particular, poate fi adevărat că

  Ion a alergat între orele 2 *şi* 3.

  deşi este posibil ca el să fi făcut şi unele popasuri de câteva minute pentru a se odihni.

  Deci, dacă un proces P acoperă un interval $T1$, este posibil ca el să acopere şi un interval $T2 \subset T1$ (dar nu este sigur !).

- Propoziţiile de completare, cum ar fi

  Ion a alergat la magazin.

  au o structură mai complexă şi pot combina mai multe forme. Putem analiza aceasta aplicând predicatele din forma logică în propoziţii mai detaliate din KRL. În particular, propoziţia anterioară poate fi utilizată într-o formulă indicând că procesul de alergare care a apărut a culminat într-o stare de existenţă a lui *Ion* la magazin, adică:

$$\exists\, T1, T2.\ ALERGA(Ion1, T1)\ \&\ LA(Ion1, MAGAZIN1, T2)\ \&\ T1 : T2$$

Tabelul 13.1 sumarizează câteva proprietăţi ale acestor clase.

Tabelul 13.1:

Clasa	**Poate fi adevărat într-un punct ?**	**Poate fi adevărat într-un interval ?**
Static	*DA*	*DA*
Activ	*NU*	*DA*
Finalizare	*DA*	*NU*
Completare	*NU*	*DA*

13.1.1 Codificarea timpului

Operatorii de timp pot fi şi ei reprezentaţi în logica temporală fără a folosi operatori modali. Ideea constă în definirea de relaţii temporale în raport cu un anumit moment de referinţă (*timpul curent*). Să presupunem că $ACUM1$ este o constantă care reprezintă timpul curent; atunci $TRECUT$ se poate codifica într-o formulă folosind cuantificatorul existenţial.

Exemplul 13.2 *De exemplu,*

Mircea era fericit.

se poate reprezenta în KR prin expresia:

$\exists T1 \,.\, T1 < ACUM1 \,.\, FERICIT(Mircea1, T1)$

Aceeaşi propoziţie, dar la timpul prezent:

Mircea este fericit.

dă expresia

$\exists T1 \,.\, ACUM1 \subseteq T1 \,.\, FERICIT(Mircea1, ACUM1)$

iar la viitor:

$\exists T1 \,.\, T1 > ACUM1 \,.\, FERICIT(Mircea1, T1)$

De remarcat că există o ambiguitate în reprezentarea timpului, deoarece unele propoziţii scrise la prezent se referă la viitor (cum ar fi *Autobuzul soseşte la amiază*), iar unele scrise la viitor se referă la prezent (*De acum el va fi în clasă*). Nu vom lua în consideraţie însă asemenea disfuncţionalităţi.

Dar chiar şi fără ambiguitate, apar unele probleme dificile. De exemplu, există mai multe moduri diferite de a arăta că ceva a fost adevărat în trecut (corespunzător imperfectului, perfectului simplu, perfectului compus şi mai mult ca perfectului). Cum să facem deosebirea între:

Maria citea anunţul.
Maria citi anunţul.
Maria a citit anunţul.
Maria citise anunţul.

Ca propoziţii simple, acest lucru este foarte dificil de realizat. Să folosim două din aceste forme în propoziţii mai complexe:

Când Virgil a deschis uşa, Maria citi anunţul.
Când Virgil a deschis uşa, Maria citise anunţul.

În prima propoziţie, actul citirii este simultan cu cel al deschiderii uşii, în timp ce în a doua propoziţie, el îl precede.

Formalizarea se bazează pe noţiunea de *timp de referinţă*.

În cele două propoziţii anterioare, timpul de referinţă este momentul în care Virgil deschide uşa. Perfectul simplu este simultan (contemporan) cu timpul de referinţă, iar mai mult ca perfectul este anterior lui. Modelul - dezvoltat de Reichenbach (1947) - foloseşte informaţii despre 3 timpi:

S - timpul vorbirii;
E - timpul evenimentului (stării);
R - timpul de referinţă.

Timpul de referinţă poate fi folosit ca adverb de timp (vezi exemplele anterioare) sau determinat de contextul discursului.

Pentru timpii simpli (perfect simplu, viitor simplu) timpul de referinţă şi cel al evenimentului coincid ($E = R$).

Exemplul 13.3 *Cele trei forme generate în acest caz de relația dintre S și R sunt exemplificate prin:*

Pisica toarce.	*Prezentul simplu:*	$S = R,\ E = R$
Pisica toarse.	*Perfectul simplu:*	$R < S,\ E = R$
Pisica va toarce.	*Viitorul simplu:*	$S < R,\ E = R$

La timpurile perfecte (perfectul compus, mai mult ca perfectul, viitorul perfect) timpul evenimentului precede timpul de referință.

Exemplul 13.4 *Cele trei situații posibile sunt exemplificate astfel:*

Pisica a tors.	*Perfectul compus:*	$S = R,\ E < R$
Pisica torsese.	*Mai mult ca perfectul:*	$R < S,\ E < R$
Pisica va fi tors.	*Viitorul compus:*	$S < R,\ E < R$

Analiza pune în evidență și alte trei posibilități, datorate relației $R < S$ (timpul posterior):

Exemplul 13.5

Pisica urmează să toarcă.	*Prezentul posterior:*	$S = R,\ R < E$
Pisica urma să toarcă.	*Trecutul posterior:*	$R < S,\ R < E$
Pisica va urma să toarcă.	*Viitorul posterior:*	$S < R,\ R < E$

13.2 Deducția automată

Să studiem în continuare câteva strategii și tehnici de raționament folosite în sistemele de reprezentare a cunoașterii. După cum s-a arătat în prelegerea anterioară, sistemele de raționament se împart în două categorii mari: ***declarative*** și ***procedurale***. Ne vom îndrepta atenția asupra câtorva tehnici pur declarative.

Una din complicațiile demonstrării automate provine din faptul că anumite formule se pot demonstra în mai multe moduri; numărul demonstrațiilor se reduce însă mult dacă se realizează o reducere a complexității formulelor înainte de demonstrație. În particular, este util de folosit o formă normală pentru formule, bazată pe un subset al sintaxei *FOPC*. De exemplu,

$$\neg P \& Q, \quad \neg(P \vee \neg Q), \quad \neg(P \supset Q)$$

sunt logic echivalente și conduc la aceeași formă normală.

Să dezvoltăm deci o formă normală pentru *FOPC*, numită ***forma normală conjunctivă (FNC)***. La nivelul constantelor, funcțiilor și propozițiilor atomice, *FNC* este identică cu cea din *FOPC*. Un literal corespunde unei propoziții atomice, posibil negată. De exemplu:

Persoană(Ion1).
Mașină(Mașină1).
Aparține(Ion1,Mașină 1).
$\neg$ *Fericit(Ion1)*

O *clauză* este o formulă având forma generală

$$(P_1, \ldots, P_n \leftarrow Q_1, \ldots, Q_m)$$

unde P_i şi Q_i sunt literali, cu semnificaţia:

Dacă $Q_1, \ldots, Q_m$ sunt adevărate, atunci cel puţin una din propoziţiile $P_1, \ldots, P_n$ este adevărată.

Exemplul 13.6 *Clauza*

(Aparţine(Elena1,Maşină2) ← Fericit(Elena1))

corespunde formulei - scrisă ca disjuncţie:

Aparţine(Elena1,Maşină2) $\vee$ $\neg$*Fericit(Elena1)*

Sub această formă, toţi Q_i sunt literali scrişi cu negaţie.

Se poate arăta că toate formulele din FOPC au o formă clauzală echivalentă. Tabelul 13.2 conţine câteva astfel de forme echivalente.

Tabelul 13.2:

Formula în FOPC	Forma clauzală echivalentă
	$(P \leftarrow)$
$\neg P$	$(\leftarrow P)$
$P \& Q$	două clauze : $(P \leftarrow)$, $(Q \leftarrow)$
$P \vee Q$	$(PQ \leftarrow)$
$Q \vee \neg P$	$(Q \leftarrow P)$
$P \supset Q$	$(Q \leftarrow P)$
$\neg(P \supset Q)$	două clauze : $(P \leftarrow)$, $(\leftarrow Q)$
$(P \& Q) \vee R$	două clauze : $(PR \leftarrow)$, $(QR \leftarrow)$

Dacă $n = 1$ avem *clauza Horn.* Strategia folosită de limbajul *Prolog*, bazată pe clauze Horn, este capabilă să demonstreze orice formulă rezultată logic din KB.

Raţionamentul cu clauze face posibilă extensia algoritmilor de unificare. Să considerăm întâi un KB limitat numai la literali (similar unei baze de date relaţionale care foloseşte cuantificatori). Aici, formula *FOPC*

$\forall y \exists x P(x, y)$

corespunde literalului

$P(Sk2(?y),\ ?y)$.

Presupunem că KB conţine acest literal şi să vedem ce se întâmplă dacă în acelaşi KB va rezulta ulterior formula

$\forall w \exists z P(w, z)$.

Transformată direct în forma clauzală, se ajunge la ceva de tipul

$P(Sk3(?w),\ ?w)$

care nu se unifică cu literalul din baza de date pentru că funcţiile Skolem sunt notate diferit. Într-un sistem bazat pe căutarea patternurilor, această problemă

se evită inversând interpretarea cuantificatorilor nedefiniţi (folosiţi cu ?). Deci, o formulă

$\forall y \exists x P(x, y)$

va conduce la

$P(?z, Sk4)$,

care se va unifica cu literalul pentru aceeaşi formulă, existent în baza de cunoştinţe.

Această idee rezultă din strategia de demonstrare automată. În general există două metode generale de verificare a unei formule P.

- Prima se bazează pe o demonstrare a lui P direct din KB, folosind regulile de inferenţă.
- A doua (tehnica *refuzului*) constă în a arăta că $\neg P$ este inconsistentă în KB (şi deci P trebuie să rezulte din KB); aceasta este modalitatea uzuală folosită în deducţia automată şi formează fundamentul teoretic al tehnicii de regăsire a patternului descris anterior, a strategiilor de demonstrare bazate pe clauze Horn.

Ambele tehnici pot fi considerate implementări specializate ale unei singure reguli de inferenţă numită *regula de rezoluţie.*

Exemplul 13.7 *Un caz foarte simplu de regulă de rezoluţie este modus ponens. În particular, fiind dată o clauză $(Q \leftarrow P)$ (adică $P \supset Q$) şi clauza $(P \leftarrow)$ (adică "P este adevărată"), regula de rezoluţie decide $(Q \leftarrow)$ (adică "Q este adevărată").*

Un alt caz simplu de regulă de rezoluţie tratează contradicţiile. Din clauzele $(P \leftarrow)$, $(\leftarrow P)$ (adică "P este adevărată" respectiv " falsă") rezultă clauza vidă $(\leftarrow)$ care indică inconsistenţa bazei de date.

Regula de rezoluţie este generalizată la *FOPC* folosind unificarea pentru a instanţia variabilele cu valori identice din cele două clauze.

Exemplul 13.8 *Să considerăm un KB care include clauze ce afirmă că toţi câinii muşcă şi că Grivei este câine:*

(MUŞCA(?x) ← CÂINE(?x))
(CÂINE(Grivei1) ←)

Regula de rezoluţie, după substituţia lui ?x cu Grivei1, va elimina literalul CÂINE(Grivei1) din cele două clauze, conducând la clauza finală (MUŞCA(Grivei1)←).

Cu aceste idei se poate determina strategia de construcţie a demonstraţiilor. Fiind dat un KB în formă clauzală, pentru a determina dacă o formulă P rezultă din KB, se procedează astfel:

- $\neg P$ se scrie sub formă de clauză şi se adaugă la KB; fie KB' noua bază de cunoştinţe.
- Folosind regula de rezoluţie în KB', se obţine clauza vidă; deci KB' este inconsistentă.

Tabelul 13.3:

(ZBOR F1)	*(STIMP F1 ESTE 17:00 HR)*
(ZBOR F2)	*(STIMP F2 ESTE 10:00 HR)*
(ZBOR F3)	*(STIMP F3 ESTE 9:00 HR)*
(ZBOR F4)	*(STIMP F4 CJ 17:00 HR)*
(AERP B)	*(PTIMP F1 CJ 16:00 HR)*
(AERP CJ)	*(PTIMP F2 B 9:00 HR)*
(AERP B)	*(PTIMP F3 CJ 8:00 HR)*

13.3 Semantici procedurale şi chestionare

Tehnicile procedurale sunt frecvent folosite în aplicaţii bazate pe chestionarea bazelor de date. În formalismul definit anterior, *KB* (baza de date) va consta numai din literali pozitivi, adesea fără variabile. În loc de a converti limbajul formei logice într-un *FOPC* extins, forma logică este tratată ca o expresie într-un chestionar. Fiecare construcţie corespunde unei proceduri particulare care rezolvă o anumită întrebare. De exemplu, întrebarea:

Pe toate zborurile spre Iaşi se serveşte masa ?

cu forma logică:

(ORICE **f1***:(&(ZBOR* **f1***)(DEST* **f1***(NUME* **c1** *"Iaşi"))) ((SERV_MASA* **f1***))*

va fi interpretat ca procedură astfel:

1. Se află toate zborurile din baza de date cu destinaţia *IS* (simbolul pentru *Iaşi*);
2. Pentru fiecare zbor aflat, controlează dacă se serveşte masa; dacă răspunsul este afirmativ pentru toate zborurile, rezultă *DA*; altfel *NU*.

Să vedem cum se pot interpreta ca proceduri expresiile formei logice; vom face referinţă la o metodă de interpretare numită ***semantică procedurală***, utilizată pe scară largă şi în modulul de analiză semantică al compilatoarelor.

Exemplul 13.9 *Vom folosi o bază de date simplă prezentată în Tabelul 13.3 Ea constă dintr-un set de literali pozitivi, fără variabile. Timpii sunt indicaţi în notaţia internaţională:* **17:00 HR** *înseamnă ora* **17:00 PM**. *Relaţia*

(STIMP f c t)

indică faptul că zborul **f** *soseşte la aeroportul* **c** *la timpul* **t***;*

(PTIMP f c t)

indică faptul că zborul **f** *pleacă din aeroportul* **c** *la timpul* **t**.

Pentru dezvoltarea chestionarului se folosesc două funcţii:

$$(Test\ < literal >_1, \ldots, < literal >_n)$$

care întoarce "Adevărat" dacă există o instanţiere a variabilelor astfel încât fiecare literal este găsit în baza de date;

$$(Află < var > < literal >_1, \ldots, < literal >_n)$$

care, în caz de succes întoarce fiecare apariţie a variabilei indicate care este o soluţie.

Astfel, pentru baza de date de sus, întrebarea

$$(Află\, ?x\ (ZBOR\, ?x)(STIMP\, ?x\ ESTE\ 10:00\ HR))$$

va întoarce valoarea **(F2)**, *deoarece F2 este singura instanţiere a lui ?x în care ambii literali sunt în baza de date.*

Toate expresiile din limbajul formei logice trebuiesc interpretate într-un mod care să se reducă eventual la cele două forme de întrebare din exemplul 13.9. Acesta este dat aplicând forma logică într-o procedură care realizează întrebările corespunzătoare din baza de date. Deci răspunsul la o întrebare este dat în doi paşi:

- Translatarea formei logice într-un program;
- Executarea acelui program pentru calculul răspunsului.

Să considerăm primul pas (cel de translatare). Pentru orice expresie logică E notăm *T(E)* translatarea lui E în limbajul de chestionar al bazei de date. Modul de translatare variază după tipul aplicaţiei. De exemplu, expresii ca *(NUME c1 "Iaşi")* sunt translatate în constante ale bazei de date, în acest caz *IS*; în plus, simbolul *c1* va fi stocat împreună cu constanta *IS* într-o *tabelă de simboluri*. Astfel, va fi posibil ca la apariţia lui *c1* într-o altă parte a formei logice, acest simbol să fie înlocuit de valoarea sa *IS*.

Unele relaţii ale formei logice se translatează direct în relaţii ale bazei de date; pentru altele sunt necesare expresii mai complexe. De exemplu, relaţia *DEST* din forma logică nu apare în baza de date; ea este codificată în relaţia *STIMP* care include două elemente: destinaţia zborului şi timpul de aterizare. Deci relaţia din forma logică

(DEST f1 (NUME c1 "Iaşi")

unde *f1* a fost deja asociat cu o variabilă *?f*, se translatează în relaţia din baza de date

(STIMP ?f ESTE ?t).

Deoarece timpul nu este conţinut în relaţia *DEST*, el este interpretat ca o variabilă fără restricţii în translatare. În general translatarea fiecărei relaţii din forma logică în baza de date trebuie specificată distinct.

Semantica procedurală are o modalitate specifică de interpretare a conectorilor logici şi a cuantificatorilor, care - bineînţeles - nu au corespondent în baza de date relaţională. Operatorii logici sunt interpretaţi astfel:

- Conjuncţii:

  $(AND\ R_1, \ldots, R_n)$

  se translatează într-un program de forma

$(VERIFIC_TOT_ADEV \quad T(R_1), \ldots, T(R_n))$

care la execuţie va verifica fiecare $T(R_i)$ dacă este adevărat şi trece variabila instanţiată întrebărilor care urmează. Dacă există un set de instanţieri de variabile cu pentru care toţi $T(R_i)$ sunt valabili, atunci programul are succes; altfel eşuează.

- Disjuncţii:

  $(OR\ R_1, \ldots, R_n)$

  se translatează într-un program de forma

  $(AFLĂ_UNUL_ADEV \quad T(R_1), \ldots, T(R_n))$

  care la execuţie va întreba succesiv fiecare $T(R_i)$ până găseşte un R_i afirmativ: în acest caz programul are succes.

- Procedura pentru negaţii presupune ipoteza universului minim la toate relaţiile bazei de date:

  $(NOT\ R)$

  se translatează într un program de forma

  $(DACĂ_NU \quad T(R))$

  care are succes numai dacă $T(R)$ eşuează la întrebare.

Cuantificatorii necesită folosirea unor transformări mai complexe. Fiecare cuantificator se translatează într-un program care efectuează o anumită operaţie pe baza de date. Din cauza limitărilor impuse de limbajul bazei de date, de obicei se pot prelucra numai citirile distributive ale cuantificatorilor multipli.

Să considerăm trei cuantificatori folosiţi în aplicaţiile de tip *întrebare/răspuns*:

- $(\mathbf{L}\ \mathbf{x} : \mathbf{R_x}\ \mathbf{P_x})$

  se translatează într-un program

  (AFLĂ_L $?x$ $T(R_{?x})$ $T(P_{?x})$)

  care află toate variabilele $?x$ care satisfac $T(R_{?x})$, adică (AFLĂ $?x$ $T(R_{?x})$).

  - Dacă se găseşte un singur răspuns, atunci acel răspuns este substituit lui $?x$ în toată expresia şi se execută $T(P_{?x})$ pentru a găsi răspunsul final la întrebare.
  - Dacă nu se găseşte nici un obiect la întrebarea adresată lui $T(R_{?x})$, atunci există o anumită contradicţie care va fi folosită în sistemul *întrebare/răspuns*; de exemplu se anunţă utilizatorul că nu există nici un obiect solicitat.
  - Dacă se găsesc mai multe răspunsuri, construcţia sistemului va decide calea care va fi urmată: unele sisteme execută $T(P_{?x})$ pentru fiecare valoare găsită, altele anunţă eroare, etc.

- (**FIECARE x : R_x P_x**)

  se translatează într-un program

  (ITERARE $?x$ $T(R_{?x})$ $T(P_{?x})$)

  care începe de asemenea prin a căuta toţi $?x$ care satisfac $T(R_{?x})$; în a doua fază va executa $T(P_{?x})$ pentru fiecare valoare găsită şi finalizează cu succes numai dacă toate aceste întrebări au răspuns acceptabil.

- (**CE x : R_x P_x**)

  se translatează într-un program

  (SCRIE_TOT $?x$ $T(R_{?x})$ $T(P_{?x})$)

  care află toate obiectele ce satisfac *(AFLĂ $?x$ $T(R_{?x})$ $T(P_{?x})$)*, după care scrie rezultatele. Determinarea celui mai bun format de scriere, în special a informaţiei suplimentare necesare, este o problemă dificilă pe care momentan o vom evita.

Plecând de la baza de date din Tabelul 13.3, putem construi câteva exemple de chestionare.

Exemplul 13.10 *Întrebarea*

Ce zbor spre Iaşi pleacă la ora 4 PM ?

va avea forma logică:

(*CE* **f1** : (*AND* (*ZBOR* **f1**)(*DEST* **f1** (*NUME* **c1**"*Iaşi*")))
(*PLECA* **t1** (*NUME* **t1** "4 *PM*")))

Ea se translatează într-o întrebare de forma:

(*SCRIE_TOT* ?*f* (*ZBOR* ?*f*)(*STIMP* ?*f* *ESTE* ?*t*)
(*PTIMP* ?*f* ?*s* 16 : 00 *HR*))

Aici relaţia DEST se aplică în relaţia STIMP descrisă anterior, iar predicatul PLECA în relaţia PTIMP. De remarcat că locul de plecare nu a fost specificat în forma logică; de aceea aici este folosit ca o variabilă. Într-o aplicaţie reală, oraşul de plecare va fi determinat de context sau prin lipsă. Pentru baza de date din Tabelul 13.3, există un singur zbor care satisface descrierea curentă, anume $F1$.

Exemplul 13.11 *Solicitarea:*

Daţi timpul de plecare al fiecărui zbor spre Iaşi.

are forma logică:

(*FIECARE* **f1** : (*AND* (*ZBOR* **f1**)(*DEST* **f1** (*NUME* **c1** "*Iaşi*")))
(*L* **t1** : (*PLEC_TIMP* **f1 t1**)(*DA_SCRIE*1 *g*1)))

Ea se transcrie în întrebarea:

(*ITERARE* ?*f*1 (*CONTROL_TOT*(*ZBOR* ?*f*1)(*STIMP* ?*f*(*ESTE* ?*t*1))
(*AFLA_L* ?*t*1 (*PTIMP* ?*f*1 ?*ORAŞ* ?*t*1)(*SCRIE* ?*t*1)))

În acest caz, interpretarea verbului "a da" foloseşte tipărirea argumentului său, adică timpul de plecare. Execuţia expresiei va parcurge următoarele etape:

1 *Se lansează prima parte a pasului ITERARE pentru a afla toţi ?f1 care verifică restricţia. Procedura CONTROL_TOT satisface verificările pentru ?f1 numai dacă (ZBOR ?f1) şi (STIMP ?f1 ESTE ?t1) sunt în baza de date. Se întorc valorile zborurilor F1, F2, F3*

2 *Se lansează a doua parte a pasului pentru a executa*

$(AFLA_L\ ?t1\ (PTIMP\ ?f1\ ?ORAŞ\ ?t1)(SCRIE\ ?t1))$

pentru fiecare din cele trei valori. Să parcurgem această execuţie cu prima valoare F1. Expresia este:

$(AFLA_L\ ?t1\ (PTIMP\ F1\ ?ORAŞ\ ?t1)(SCRIE\ ?t1))$

Programul pentru AFLA_L realizează întâi întrebarea
$(AFLA\ ?t1\ (PTIMP\ f1\ ?ORAŞ\ ?t1))$.
Este întors un singur răspuns, anume timpul 17 : 00 *HR*.
Al doilea pas din programul AFLA_L execută SCRIE pentru această valoare tipărind valoarea 17 : 00 *HR*.
A doua şi a treia iteraţie vor tipări valorile 10 : 00 *HR respectiv* 9 : 00 *HR*

13.3.1 LUNAR - Un sistem de chestionare bazat pe o bază de date limbaj natural

La începutul anilor '70 a fost dezvoltat un sistem de chestionare numit *LUNAR*, bazat în mare parte pe ideile prezentate în paragraful anterior. Baza de date construită conţinea informaţii despre mostrele de sol lunar aduse de pe Lună de misiunile *Apollo*. Acesta a fost primul sistem bazat pe limbaj natural care a aplicat în practică construcţiile teoretice; el a fost de asemenea folosit ulterior şi în alte sisteme.

LUNAR foloseşte un analizor *ATN* care produce o reprezentare bazată pe relaţii gramaticale; ea este interpretată cu un modul interpretor semantic care foloseşte o tehnică de căutare realizând o expresie în limbajul de interpretare. Informaţia dată de cuantificatori este păstrată şi prelucrată separat de restul reprezentării semantice, prin procedee euristice. Rezultatul este o reprezentare a înţelesului, exprimată într-un limbaj similar celui construit aici, plecând de la limbajul formei logice. Execuţia se realizează folosind semantica procedurală.

Câteva exemple de cerinţe pe care proiectul *LUNAR* le poate îndeplini:

Găseşte toate mostrele lunare cu proprietăţi magnetice.
În care roci a fost identificată apatită ?
Care este activitatea specifică a lui $A - 126$ *în sol ?*
Care este concentraţia medie de olivină în probele din zona $X23$ *?*
În care roci concentraţia medie de titan depăşeşte 6% *?*

Prelegerea 14

Contextul local al discursului

14.1 Componentele contextului local al discursului

Definiţia 14.1 *Contextul local al discursului cuprinde sintaxa şi semantica propoziţiilor care preced propoziţia curentă, împreună cu o listă a obiectelor menţionate în propoziţie, la care se poate face referire ulterior prin pronume sau construcţii substantivale.*

Acest context local este util în multe situaţii. De exemplu el poate conţine antecedente ale pronumelor, ca în:

1a. Sorin$_i$ şi-a pierdut portofelul$_j$ în maşină.
1b. El$_i$ l$_j$-a căutat mult timp.

De asemenea, contextul local ajută la interpretarea propoziţiilor care folosesc construcţii verbale eliptice. Exemplu:

2a. Sorin şi-a uitat portofelul.
2b. La fel şi Tudor.

Construcţiile de acest gen se referă în mod normal la evenimentele descrise de propoziţiile imediat anterioare. Astfel, în

3a. Sorin şi-a uitat portofelul.
3b. El a căutat pe cineva să-l împrumute cu bani.
3c. La fel şi Tudor.

propoziţia *3c.* nu semnifică faptul că Tudor şi-a pierdut şi el portofelul.

Există însă probleme privind lucrul cu propoziţiile în context local; acestea sunt provocate în special de prezenţa conjuncţiilor. De exemplu, propoziţiile *3a* şi *3b* pot fi unite în una singură folosind o conjuncţie, dar asta nu schimbă înţelesul final.

4a. Sorin şi-a uitat portofelul, aşa că a căutat pe cineva să-l împrumute cu bani.
4b. La fel şi Tudor.

Este greu de interpretat *4b* sub forma că şi Tudor şi-a pierdut portofelul.

De menţionat că o propoziţie simplă care foloseşte o conjuncţie, acceptă expresii verbale eliptice cum ar fi:

5. Sorin a plecat de la curs, ca şi Tudor.

În schimb, o conjuncţie aşezată între expresii verbale conduce în caz de elipsă, la păstrarea întregului context. Deci:

6a. Sorin şi-a uitat portofelul şi a pierdut toţi banii.
6b. La fel şi Tudor.

În acest caz, Tudor a păţit aceleaşi necazuri: uitat portofel, pierdut banii.

Din aceste exemple, putem deduce contextul local ca fiind derivat din clauzele principale precedente (nu din propoziţiile precedente !). Astfel, deoarece conjuncţiile (cum ar fi *şi*) pot alipi clauze majore (vezi exemplul 5), se poate afirma:

Prima parte a conjuncţiei stabileşte contextul local pentru a doua parte.

Conjuncţiile între *VP*-uri apar în interiorul unei clauze majore, aşa că ambele au acelaşi context local (*6a* şi *6b*). Clauzele secundare sunt incluse drept componente ale unor clauze majore, deci ele nu vor crea un context local nou. Putem exemplifica aceasta prin:

7a. Sorin şi-a uitat portofelul când s-a dus la cinema.
7b. La fel şi Tudor.

Propoziţia *7b* poate fi interpretată că Tudor şi-a pierdut portofelul (eventual când s-a dus la cinema), dar nu că Tudor s-a dus la cinema.

O parte importantă a contextului local este lista antecedentelor posibile pentru pronume; o vom numi *lista entităţilor discursului (DE)*. Lista *DE* este un set de constante definite în *KB*, care reprezintă obiectele menţionate în ultima clauză majoră şi la care se poate face referire prin pronume. Chiar dacă o *DE* nu este menţionată explicit în clauza precedentă, ea este introdusă implicit. Pentru a rezolva astfel de situaţii vom vorbi despre obiectele pe care *le evocă* (sau la care *face referire*) o propoziţie (obiectele menţionate explicit sau implicit de aceasta).

Când un pronume are un anumit antecedent, se înţelege că atât pronumele cât şi antecedentul său se referă la acelaşi obiect; spunem că pronumele şi antecedentul său *co-referă*. De remarcat că un pronume şi antecedentul său se pot co-referi la acelaşi obiect X, chiar dacă nici vorbitorul şi nici auditoriul nu pot identifica X într-un mod concret. De exemplu:

8a. Marin şi-a cumpărat ieri o maşină$_i$.
8b. Ea$_i$ *a costat foarte mult.*

Afirmaţiile sunt făcute fără ca vre-un participant la conversaţie să fi văzut maşina sau să aibă vre-un mijloc de a o identifica. Deşi această co-referire introduce anumite probleme în reprezentare (nu se poate defini un obiect din lumea reală fără a-l identifica), o soluţie constă în folosirea de funcţii Skolem sub o formă similară celei din Prelegerea 12: adăugarea de noi termeni la limbaj.

Exemplul 14.1 *Dacă "o maşină" are forma logică*

< *O* **c1** *MAŞINĂ* >,

constanta Skolem generată va fi C1. Co-referinţa este indicată folosind egalitatea. Deci, dacă pronumele "Ea" are forma logică

(*PRO* **i1** *EA*1),

atunci faptul că "Ea" co-referă cu "C1" va fi prins în aserţiunea

(*I*1 = *C*1).

14.1.1 Entităţi de generare a discursului

Pentru fiecare expresie substantivală este generată o entitate specifică a discursului. *NP* distincte reprezintă restricţii distincte ale universului discursului. Un *NP* nedefinit evocă de obicei o nouă entitate a discursului care uzual nu mai trebuie identificată ulterior în *KB*. Un nume propriu - pe de-altă parte - descrie în general un anumit obiect din *KB* care are asociat acelaşi nume. Un *NP* definit (inclusiv pronumele) se referă la un obiect menţionat anterior în discurs, de obicei o entitate a discursului în context local. Pluralul *NP*-urilor reprezintă mulţimi de obiecte. Un *NP* complex care foloseşte conjuncţii reprezintă o mulţime formată din elementele conjuncţiei.

Exemplul 14.2 *Expresia substantivală*

Ion şi Ana.

semnifică trei DE: Ion1, Ana1 şi {Ion1 Ana1}.

În cele ce urmează vom considera entităţile discursului reprezentate prin *NP*-uri nedefinite. Această clasă este foarte importantă deoarece elementele ei introduc numeroase entităţi noi ale discursului şi acoperă multe elemente necesare prelucrării referinţelor definite (care vor fi discutate în prelegerea următoare). Pentru a calcula mulţimea entităţilor discursului, vom converti întâi forma logică în reprezentarea cuantificatorilor descrisă anterior, reducând toţi cuantificatorii limbajului natural la cei universal şi existenţial definiţi pe baza mulţimilor scrise explicit.

De asemenea, în exemplele simple folosite, vom presupune că toate numele proprii şi *NP*-urile definite sunt înlocuite cu constante din *KB* reprezentând referinţe la acestea.

Pluralul *NP*-urilor va fi notat cu aceeaşi mulţime, indiferent dacă este vorba de o interpretare colectivă sau distributivă. În citirea colectivă, mulţimea introdusă este folosită direct în propoziţie ca argument, în timp ce în citirea distributivă mulţimea reprezintă domeniul cuantificatorului universal. Pentru definirea mulţimii, modificatorii trebuiesc împărţiţi în faza de translatare în două clase:

- Restricţii ale mulţimii (SR).
- Restricţii ale fiecărui obiect al mulţimii (IR).

De exemplu, translatarea expresiei substantivale *Trei băieţi* din propoziţia

9. Trei băieţi l-au hrănit pe Grivei.

Tabelul 14.1:

Exemplu	Entitatea Discursului (DE)	Restricţia de mulţime (SR)	Restricţia individuală (IR)
un om	$M1$	*none*	$OM1(M1)$
trei femei	$W1$	$\|W1\| = 3$	$W1 \subseteq \{w \| FEMEIE1(w)\}$ sau, echivalent, $\forall w . w \in W1 \supset FEMEIE1(w)$
câteva pisici negre	$C1$	$\|C1\| > 1$	$C1 \subseteq \{c\|$PISICĂ1$(c)\&$ $NEGRU1(c)\}$

va folosi un *SR* care arată că mulţimea are trei elemente şi un *IR* care specifică că fiecare element este un băiat.

În Tabelul 14.1 sunt date translatări pentru câteva *NP* nedefinite.

14.1.2 *NP* nedefinite peste domeniul cuantificatorilor universali

Dacă un *NP* apare - în urma unei interpretări distributive - în domeniul unui cuantificator universal, se ivesc dificultăţi destul de mari. Astfel, o expresie substantivală nedefinită - la singular - folosită în domeniul unui cuantificator universal, poate reprezenta atât o mulţime cât şi un element.

Să considerăm următorul discurs:

10a. Trei băieţi$_i$ cumpără câte o pizza$_j$.
10b. Ei$_i$ le$_j$-au mâncat în parc.

De remarcat că

Ei au mâncat-o în parc

nu este o continuare corectă a propoziţiei *10a*. Ce proprietăţi sunt necesare pentru a defini entitatea discursului generată în *10a* de ***o pizza*** ? Mulţimea definită prin $\{x|PIZZA(x)\}$ este prea generală, pentru că ***le$_j$-au*** din ***10b*** nu se referă la mulţimea tuturor pizz-elor. O tratare mai corectă constă în definirea unei noi entităţi *P1* a discursului, reprezentând o submulţime a lui $\{x|PIZZA(x)\}$; în acest fel, ***le$_j$-au*** din *10b* se poate referi la o mulţime corectă de pizze, anume acele introduse de ultima propoziţie.

Această tratare pierde totuşi informaţie despre *P1*, în particular faptul că mulţimea constă din pizz-ele cumpărate de băieţii menţionaţi la *10a.* De aceea este preferabil să se îmbogăţească reprezentarea lui *P1* considerând formula originală pentru *10a*:

$\forall b : b \in B1 . \exists p : PIZZA(p) . CUMPĂRĂ(b, p)$

unde $|B1| = 3 \ \& \ B1 \subseteq \{x|BĂIAT(x)\}$.

Forma skolemizată este:

$\forall b : b \in B1 . PIZZA(sk4(b)) \ \& \ CUMPĂRĂ(b, sk4(b))$

unde $sk4(b)$ este o funcţie nouă reprezentând pizza cumpărată de fiecare băiat. Analiza completă a propoziţiei este:

Propoziţie: *Trei băieţi au cumpărat câte o pizza.*
Translatare iniţială:
$\exists B : |B| = 3 \ \& \ B \subset \{x|BĂIAT(x)\}$
$\forall b : b \in B \ . \ \exists p : PIZZA(p) \ . \ CUMPĂRĂ(b,p)$
Entităţile discursului:
$B1 : |B1| = 3 \ \& \ B1 \subset \{x|BĂIAT(x)\}$
$P1 : P1 = \{x|PIZZA(x) \ \& \ \exists y : y \in B1 \ . \ x = sk4(y)\}$
Conţinutul semantic:
$\forall b : bB1 \ . \ sk4(b) \in P1 \ \& \ CUMPĂRĂ(b, sk4(b))$

14.2 Un model de referire bazat pe liste istorice

În această secţiune vom construi o tehnică simplă pentru identificarea antecedentelor pronumelor. Ea se bazează pe *liste istorice* - o stivă de liste ale entităţii discursului generate de propoziţiile anterioare; entităţile din discursul local curent (adică cele generate de clauza precedentă) sunt în topul stivei.

Să presupunem că a fost definit algoritmul de generare a entităţilor discursului produse de o propoziţie. Lista istorică va consta din toate aceste entităţi, evocate într-un trecut destul de recent. Unele sisteme reduc acest trecut la ultimele $1-3$ contexte locale, în timp ce altele nu fac nici o limitare.

Fiind dată lista istorică, algoritmul de aflare a unui antecendent operează astfel:

Se caută în cel mai recent context local un antecedent care să verifice toate restricţiile relative la pronume.

Restricţiile pot proveni din orice sursă. De exemplu, reflexivitatea poate elimina anumite obiecte din lista ascendenţilor posibili; la fel genul, numărul, şi alte condiţii. Dacă nu se găseşte nici un antecedent în contextul local curent, se activează lista istorică a următorului context local şi procedeul de căutare se reia.

Algoritmul acesta poartă numele de *restricţia recentă* şi are drept conjectură faptul **că antecedentul trebuie să fie cel mai recent obiect menţionat care satisface toate restricţiile.**

Exemplul 14.3 *Să considerăm următorul discurs, referitor la o cursă maritimă:*

11a. Antreprenorii aveau mulţi bani şi au cheltuit mult pentru a construi nava de concurs.

11b. Băieţii în schimb şi-au construit nava cu un buget mult mai mic.

11c. Ei ştiau că vor câştiga cursa fără probleme.

Acest "Ei" din 11c se referă mai degrabă la băieţi, deşi - la o analiză semantică mai atentă - este mai probabil că antreprenorii sunt cei care aveau convingerea câştigării cursei.

Tabelul 14.2:

Propoziţie	Generarea entităţilor discursului
11*b*	$B2: B2 \subseteq \{x \mid BĂIAT(x)\}$
	$B3:$ $NAVĂ(B3)$ & $CONSTRUIT(B3, B2)$
	$B4:$ $BUGET(B4)$ & $FOND_LIMITAT(B3)$
11*a*	$C1: C1 \subseteq \{a \mid ANTREPRENOR(a)\}$
	$M1:$ $BANI(M1)$
	$B1:$ $NAVĂ(B1)$ & $PROPRIETAR(B1, C1)$

Lista istorică a propoziţiei 11c *poate arăta ca în Tabelul 14.2. Pentru a afla antecedentul pronumelui "Ei" trebuie căutat în această listă primul obiect care satisface restricţiile pronumelui. Deci trebuie căutat un obiect x care satisface condiţiile "Ei(x)", unde Ei(x) este adevărat dacă:*

- *x reprezintă o fiinţă;*
- *x este la plural.*

Prima entitate testată, $B2$ *satisface aceste condiţii şi va fi selectată.*

Aceeaşi tehnică se poate folosi pentru descrieri definite. De exemplu, dacă 11c *este*

Ei ştiau că nava lor va învinge uşor.

antecedentul expresiei substantivale "nava" va fi primul obiect x care satisface restricţia NAVA(x), care va fi $B3$.

14.3 Pronume şi centralizare (Focus)

Restricţia recentă stabileşte deci o ordine de prioritate pe mulţimea contextelor locale. Problema care se pune este aceea de a realiza o relaţie de ordine şi în interiorul unui context local. Se pare că limbajul natural acordă o anumită preferinţă obiectelor care ocupă roluri centrale în clauzele majore peste entităţile discursului. În anumite condiţii, aceste preferinţe sunt destul de tari pentru a interfera cu interpetările logice posibile.

Să considerăm discursul:

Maria a vărsat berea pe masă. Ea era rece.

Chiar dacă restricţia recentă şi cea semantică sugerează că *Ea* se referă la *masă*, mulţi au îndoieli în această interpretare şi consideră că de fapt este vorba de *bere*.

Preferinţele între argumentele majore din clauza principală sunt mai subtile. Astfel:

12a. Sandu l-a văzut pe Andrei la petrecere.

12b. El s-a dus la bar şi şi-a luat altă băutură.

Deşi atât Sandu cât şi Andrei sunt antecedenţi posibili pentru pronumele *El* din 12*b*, există se pare o preferinţă pentru Sandu. Această preferinţă nu este atât de puternică încât să acopere influenţele semantice şi contextuale, cum ar fi în:

13a. Sandu l-a văzut pe Andrei la petrecere.
13b. Băuse prea mult.

Aici este mai greu de interpretat că propoziţia 13*b* se referă la Sandu, şi nu este evident că cele două propoziţii sunt legate una de alta.

Există se pare un factor care influenţează procesul de interpretare; el se bazează pe noţiunea de *centru* sau *focalizarea discursului.* În general discursurile sunt organizate în jurul unui obiect despre care se ocupă discursul. Acest obiect, numit *centru*, tinde să rămână acelaşi pentru un anumit număr de propoziţii, după care locul lui este ocupat de alt obiect. De asemenea se observă tendinţa ca centrul unei propoziţii să fie pronominalizat.

Aceasta afectează interpretarea pronumelor, deoarece odată stabilit un centru, va fi o tendinţă puternică ca pronumele secundare să continue să se refere la el. De exemplu:

14a. Radu a plecat târziu la petrecere.
14b. Când el a ajuns, Sorin l-a întâlnit la uşă.
14c. El a decis să plece devreme.

Semantic, 14*c* are sens cu oricare din cei doi subiecţi (*Radu* şi *Sorin*) ca antecedent; preferinţa structurală este pentru *Sorin*, deoarece acesta joacă un rol central în clauza majoră din *14b*. Teoria centrului totuşi îl dă pe *Radu* ca antecedent deoarece în 14*b* se face referire la el prin pronume - deci *Radu* este centrul în 14*b*, iar nimic din 14*c* nu indică faptul că centrul s-ar fi schimbat.

Mai precis, în teoria centrului sunt folosite două structuri interactive:

- Entităţile discursului în contextul local, pe care le numim *următorii centri potenţiali*; aceştia sunt listaţi într-o ordine care reflectă preferinţele structurale: subiectul, apoi obiectele directe, obiectele indirecte, iar la sfârşit celelalte entităţi ale discursului din propoziţie. Primul din listă este numit *următorul centru preferat* (**Cp**).

- Centrul, notat cu **Cb** (*centrul anterior definit*), la care se referă propoziţia curentă. *Cb* este unul din următorii centri potenţiali şi este pronominalizat.

Restricţiile între centru şi pronominalizare pot fi enunţate astfel:

- **Restricţia de centralizare 1**:

  Dacă în propoziţia curentă se face referire printr-un pronume la un obiect din contextul local, atunci centrul propoziţiei trebuie să fie şi el pronominalizat.

- **Restricţia de centralizare 2**:

  Centrul trebuie să fie entitatea din discurs preferată în contextul local, la care se face referire printr-un pronume.

- **Restricţia de centralizare 3**:

  La trecerea de la o propoziţie la următoarea, păstrarea centrului are prioritate faţă de schimbarea lui.

Relativ la restricţia 1, de remarcat că dacă în propoziţie există numai un pronume, acesta va identifica centrul în mod neambiguu. În a doua restricţie, dacă şi a doua propoziţie conţine un singur pronume şi este contextual rezonabil ca cele două pronume să se refere la acelaşi obiect, atunci ele vor co-referi.

Exemplul 14.4 *Să considerăm discursul:*

15a. Paul$_1$ l$_2$-a văzut în parcare$_3$.
15b. El$_4$ se urca pe bicicletă$_5$.

Vom nota DR_1, DR_2, DR_3 cele trei entităţi ale discursului generate respectiv de "Paul", "l", "parcare". DR_2 este centrul (Cb) propoziţiei 15a. Lista următoarelor centre potenţiale este, în ordine DR_1, DR_3; primul element din această listă, DR_1 este preferat (Cp) ca următor centru. În acest context local, să considerăm propoziţia 15b. Din motive pur semantice, ea se va referi la DR_1 sau DR_2. Restricţia de centralizare oferă ca preferinţă $DR_4 = DR_2$, continuând cu acelaşi centru. Din acest motiv, DR_2 este centrul lui 15b, aşa cum a fost şi al lui 15a.

O propoziţie cu mai multe pronume este mai complexă din cauza posibilelor ambiguităţi relative la pronumele care va corespunde centrului. Astfel, restricţia de centralizare dată mai jos arată că nu există nici o preferinţă în următorul exemplu.

Exemplul 14.5 *Să considerăm următorul discurs:*

16a. Nicolae$_1$ se plimba prin parc$_2$ când l$_1$-a întâlnit pe Gabriel$_3$.
16b. El$_4$ l$_5$-a invitat la o petrecere$_6$.

Cu restricţia de centralizare 1, centrul lui 16a este fără nici o ambiguitate DR_1, adică "Nicolae" (pentru că există un singur pronume). Propoziţia 16b conţine două pronume, aşa că restricţia de centralizare poate fi satisfăcută de una din relaţiile $DR_4 = DR_1$, $DR_5 = DR_1$; nu există nici o preferinţă dacă "El$_4$" din 16b se referă la "Nicolae" sau "Gabriel". În funcţie de intuiţie, acesta este ales dintre ei; în majoritatea cazurilor este ales "Nicolae".

Să facem un studiu formal al acestei situaţii. Există patru combinaţii posibile între centrul unei propoziţii şi centrul următoareia. Ele sunt sumarizate în Tabelul 14.3. Când centrul rămâne acelaşi între cele două propoziţii, tranziţia este o continuare

Tabelul 14.3:

	$Cb_2 = Cp_2$	$Cb_2 \neq Cp_2$
$Cb_1 = Cb_2$	Continuare	Finalizare
$Cb_1 \neq Cb_2$	Schimbare cu preferinţă	Schimbare fără preferinţă

sau o finalizare a celei precedente, după cum următoarea preferinţă de centru se schimbă sau nu. Dacă centrul se schimbă, apar două cazuri, după cum noul centru este sau nu cel preferat.

Aceste restricţii pot fi formulate astfel:

- **Restricţia de centralizare 3′:**

  În Tabelul 14.3 ordinea preferinţelor este:

  Continuare - Finalizare - Schimbare cu preferinţă - Schimbare fără preferinţă.

Să considerăm acum din nou interpretarea discursului 16, prezentată schematic în Tabelul 14.4:

Tabelul 14.4:

Propoziţia 16a: *Nicolae*$_1$ *se plimba prin parc*$_2$ *când* l_1*-a întâlnit pe Gabriel*$_3$.
Entităţile discursului: $Cp:\ DR_1 = Nicolae_1$;
Altele: $DR_2 = Parc7$, $DR_3 = Gabriel1$
Centrul discursului: $Cb:\ Nicolae1$

Propoziţia 16b: *El*$_4$ l_5*-a invitat la o petrecere*$_6$.
Interpretarea 1: (Continuare)
Entităţile discursului: $Cp:\ DR_4 = Nicolae1$;
Altele: $DR_5 = Gabriel1$, $DR_6 = Petrecere1$
Centrul discursului: $Cb:\ Nicolae1$
Interpretarea 2: (Finalizare)
Entităţile discursului: $Cp:\ DR_4 = Gabriel1$;
Altele: $DR_5 = Nicolae1$, $DR_6 = Petrecere1$
Centrul discursului: $Cb:\ Nicolae1$

Deoarece Nicolae şi Gabriel sunt referiţi pronominal în 16*b*, restricţia 2 cere ca

centrul să fie *Nicolae1*. Dar aceasta nu precizează exact la cine se referă El_4; de aceea sunt două antecedente posibile: DR_1 - centrul în contextul local, indicând o continuare a centrului precedent, şi DR_3 - indicând o schimbare către următorul centru preferat. Restricţia 3′ preferă interpretarea 1, aşa că El_4 se referă tot la *Nicolae1*.

14.4 Descrieri definite

O *descriere definită* se referă la un obiect unic determinat în context.

Pentru aceasta se poate folosi informaţie din toate cele patru tipuri de context: obiectul poate fi definit unic sau prin proprietăţi specifice, sau poate fi descris în contextul local sau global al discursului.

Expresiile substantivale trebuiesc privite distinct după modul lor de definire: *existenţial* sau *referenţial*. Astfel:

- *Citirea existenţială* asigură existenţa unui obiect unic care satisface condiţiile descrierii;

- *Citirea referenţială* foloseşte descrierea pentru a se referi la un obiect cunoscut anterior, prin numele său sau printr-un pronume.

  Prin citirea referenţială, auditoriul poate identifica obiectul de referinţă, fără să fie obligat să accepte existenţa acestuia.

De remarcat că o descriere definită nu trebuie să identifice obiectul într-un sens absolut al cuvântului, aşa că se poate folosi contextul discursului pentru a restrânge aria referinţelor posibile. Uneori o expresie substantivală referenţială se poate referi cu succes la un obiect care nu este identificabil în univers de către participanţii la discuţie.

De exemplu, să considerăm discursul:

17a. Elena a cumpărat ieri o maşină$_i$ şi o rulotă.
17b. Ea a plătit prea mult pentru maşină$_i$.

El este neinteligibil dacă vorbitorul sau ascultătorul nu pot identifica maşina în lumea reală. Descrierea definită *maşină* din 17*b* identifică în mod unic un obiect din *KB*, anume acea maşină pe care *Elena* a cumpărat-o, evocată în propoziţia precedentă.

Acest exemplu sugerează o paralelă strânsă între descrierile definite şi pronume, deşi există şi unele diferenţe. Astfel, descrierile definite nu posedă acea tendinţă puternică de a avea antecedente în contextul local imediat, cum au pronumele. De asemenea, ele nu se pot co-referi cu mai mulţi constituienţi în aceeaşi propoziţie.

Descrierile definite se pot referi la obiecte care nu au fost introduse anterior în contextul discursului. Aceste obiecte sunt unice pentru discurs, în funcţie de un anumit înţeles.

De exemplu, descrierea definită *Luna* se referă în mod normal la satelitul natural al Pământului, deşi sunt numite Luni şi sateliţi ai altor planete; deci se poate face

referire la ele în anumite contexte; dar fără un context specific care face relevantă existenţa altor Luni, descrierea se referă la obiectul standard.

De asemenea, descrierile definite se pot referi la obiecte vizual prezente pentru participanţi, dar care nu au fost menţionate.

De exemplu, dacă am ajuns în faţa unei case, pot ruga pe cineva să îmi deschidă uşa (fără a face referire explicită la care uşă anume; se subînţelege că este vorba de uşa din faţă a casei).

Se pot specifica restricţiile unei referiri definite corect introducând conceptul de *unicitate contextuală.*

Definiţia 14.2 *Un obiect O este unic contextual în raport cu o descriere DX, o situaţie S care include definirea specifică şi generală şi un context al discursului format dintr-o secvenţă de contexte locale ale discursului $C_1, \ldots, C_n$ dacă şi numai dacă:*

1. *Există un număr k astfel încât:*
   (a) *pentru toţi $i < k$, nu există nici un obiect $x \in C_i$ pentru care D_x este adevărată;*
   (b) *O este singurul obiect în C_k astfel încât DX este adevărată.*
2. *Altfel, există un obiect unic O în S astfel încât DX este adevărată.*

Această definiţie sugerează un algoritm de prelucrare a expresiilor substantivale, care modifică modul de căutare prin listele istorice.

Intrare: O descriere definită de forma $< UL\ x\ D_X >$ unde D_X este formula care reprezintă descrierea.
Algoritm:

1. Se testează fiecare entitate r din contextul local al discursului pentru a vedea dacă D_r este adevărată.
   - Dacă D_r este adevărată pentru un singur r, atunci acesta este referentul.
   - Dacă se găsesc mai multe elemente r, atunci descrierea este abandonată într-un mod sau altul.
   - Dacă nici o entitate a discursului nu satisface D_X, atunci procedura se repetă pe contextul anterior al discursului.
2. Dacă nu s-a găsit nici un referent în nici un context local, se testează dacă D_X este adevărată pentru un obiect (unic) din S.

Ca la interpretarea pronumelor, decizia finală despre referent va trebui dată de sistemul general de raţionament. Deci, algoritmul menţionat mai sus va trebui să sugereze o listă total ordonată de referenţi posibili. Această informaţie va fi adăugată la forma logică, similar cum s-a procedat pentru pronume.

Pentru a identifica referenţii unei expresii substantivale complexe, toţi termenii referenţiali din expresie trebuiesc rezolvaţi simultan.

Exemplul 14.6 *Să considerăm NP "casa din colţ".*

Dacă pentru a găsi referentul la această expresie încercăm să stabilim o ordonare, pot apare dificultăţi suplimentare.

De exemplu, să presupunem că se caută întâi referentul "la colţ". Se poate imagina o situaţie în care există două colţuri P_1, P_2 şi două case C_1 - aflată în colţul P_1, şi C_2 - situată pe aceeaşi stradă S. Această situaţie poate fi descrisă de formula:

$Casa(C_1)$, $Casa(C_2)$, $Colţ(P_1)$, $Colţ(P_2)$, $Stradă(S)$
$În(C_1, P_1)$, $În(C_2, S)$.

NP "casa din colţ" se referă în mod clar la C_1, dar "colţ" este ambiguu între P_1 şi P_2.

Deci, aflarea unui referent unic pentru "colţ" va eşua, aşa că va eşua de asemenea căutarea pentru tot NP.

Căutarea simultană elimină acest neajuns; în acest exemplu, se va căuta simultan un obiect care satisface formula

$\exists c : CASĂ(c)\ \exists p : COLŢ(p)\ .\ ÎN(c, p)$

La această întrebarea răspunsul este unic, anume când c este instanţiat cu C_1 iar p cu P_1.

14.4.1 Citirea existenţială şi referinţa indirectă

Citirea existenţială identifică un obiect unic definit de descriere. Dar chiar unicitatea nu este o condiţie suficientă pentru determinarea acestui obiect.

De exemplu, dacă ne aflăm într-o şcoală şi directorul spune

Copilul este bolnav.

acesta, deşi unic, nu este determinat fără informaţii suplimentare.

De fapt, citirea existenţială poate doar introduce un obiect, definindu-l în termenii altor obiecte care pot fi definite referenţial. Fie exemplul:

18. Câştigătorul la cursa de 3000 *m obstacole a fost român.*

Nici vorbitorul şi nici auditoriul nu trebuie să ştie exact cine a câştigat cursa, dar ambii sunt de acord că acesta a fost unic, şi - mai mult - acea persoană este de cetăţenie română.

Deci, descrierile definite nu se referă totdeauna la obiecte aflate deja în istoria discursului. Ele pot însă introduce obiecte. Acest proces poartă numele de *acomodare*, datorită faptului că auditoriul acomodează vorbitorul introducând obiecte şi proprietăţi necesare pentru ca propoziţia să fie înţeleasă.

Pentru unicitatea existenţei este introdus cuantificatorul;

$$\exists! x :\ R_x\ .\ P_x$$

Această formulă este adevărată numai dacă există un obiect unic care satisface regula R_x, şi dacă P_x este adevărată pentru acest obiect.

Cu acest cuantificator, propoziţia 18 - în care presupunem că proba de 3000 m obstacole se referă la obiectul $CURSA_3000$ - se translatează în:

$$\exists! c :\ \text{CÂŞTIGA}(c, CURSA_3000)\ .\ \text{ROMÂN}(c)$$

Entităţile discursului evocate de expresiile substantivale definite existenţial sunt generate la fel ca cele nedefinite. Singura diferenţă este că aici referentul este definit precis de către descriere.

În particular, entitatea discursului pentru *câştigătorul cursei de* 3000 *m obstacole* va fi entitatea discursului C_1, unic definită prin

$$\text{CÂŞTIGA}(C_1, CURSA_3000).$$

În schimb, pentru o expresie substantivală nedefinită cum ar fi

Un alergător în proba de 3000 *m obstacole a fost român.*

entitatea discursului R_1 va fi definită prin:

$\text{ALERGĂTOR_IN}(R_1, CURSA_3000)$ & $\text{ROMÂN}(R_1)$

Unele citiri existenţiale nu includ explicit termeni referenţiali care să poată fi folosiţi la identificarea lor unică. În aceste cazuri, obiectul şi relaţia trebuiesc interferate din contextul discursului.

Ca exemplu, să considerăm discursul:

19a. Clubul la care activez a organizat o cursă.
19b. Câştigătorul a primit o maşină.

Nu există nici un complement în expresia substantivală *câştigătorul* care să indice în raport cu ce este definit el. Dar, dacă termenului lexical *câştigătorul* îi este dată aceeaşi interpretare semantică ca la 18, aceasta sugerează că interpretarea poate fi:

$$\exists! c: \ CÂŞTIGA(c, *PRO*)$$

unde **PRO** este un termen anaforic care trebuie definit prin context.

Această idee este mai clară dacă se dă un exemplu unde nu există un concurs unic determinat de context. Astfel:

20a. Atât clubul la care activez cât şi cel la care este înscris Radu au organizat curse.
20b. Câştigătorul a primit o maşină.

Propoziţia 20*b* este sau greşită (nu are nici un referent) sau foarte imprecisă deoarece suportă o interpretare în care ambele curse au avut acelaşi câştigător.

Exemple de referire indirectă pot apare folosind cuvinte care nu sunt interpretate în mod normal ca funcţii relative la un alt obiect. Astfel:

21a. Toma a adus un stilou la masă.
21b. Dar a constatat că are vârful rupt.

NP vârful din 21*b*, care - intuitiv - se referă la vârful stiloului pe care l-a adus *Toma* la masă, constituie problema. Acest tip de referinţă poate fi caracterizat definind o relaţie R şi un termen anaforic **PRO**. Interpretarea corectă a acestei expresii substantivale necesită aflarea unei forme şi a unei relaţii **R**. *Vârful* se va aplica atunci într-o citire existenţială de forma:

$$\exists! v: \ \text{VÂRF}(v) \ \& \ *R*(v, *PRO*)$$

unde **R** şi **PRO** vor fi identificate prin context. În acest exemplu, se ajunge la forma

$$\exists! v: \ \text{VÂRF}(v) \ \& \ \text{PARTE_DIN}(v, \text{STILOU})$$

unde s-a presupus că *STILOU* este entitatea discursului generată de *stilou* din 21*a*.

De asemenea, deoarece cuvântul *vârf* este ambiguu (având mai multe înţelesuri: vârf de munte, vârf de cuţit etc), relaţia definită îi identifică şi sensul corect din propoziţie.

De remarcat că unele relaţii folosite pe poziţia **R** sunt foarte uzuale şi nu aduc multă informaţie în context.

Astfel, relaţia *PARTE_DIN* este de fapt o relaţie de incluziune, des folosită. De exemplu,

Când am intrat în clasă, am observat că tabla era zgâriată.

Aici trebuie introdusă şi informaţia că în general clasele au table.

14.4.2 Referinţe la obiecte neevocate prin *NP*

Toate situaţiile de citiri referenţiale studiate în această prelegere au folosit antecedente evocate prin expresii substantivale. Sunt însă posibile referiri şi la alte obiecte ale propoziţiei. În majoritatea cazurilor, conţinutul expresiei care face referirea specifică explicit faptul că această referire este de un tip special.

În general referentul acestor construcţii este găsit în contextul local. Să considerăm de exemplu câteva continuări ale propoziţiei

Ion s-a dus la Braşov anul trecut.

Se pot face referiri la acest eveniment, ca în propoziţiile:

Călătoria i-a schimbat viaţa.
Aceasta i-a schimbat viaţa.

Se poate face referinţă la acţiunea pe care a făcut-o *Ion:*

El face acum asta în fiecare an.
Paul a reuşit abia acum.

Ne putem referi la faptul că *Ion* s-a dus la *Braşov*, prin propoziţii de forma:

Asta a luat-o prin surprindere pe Anca.

Se poate face - în sfârşit - referire la timpul şi locaţia evenimentului, folosind pronumele speciale *atunci* şi *acolo:*

Atunci era iarnă.
Îi place să se ducă acolo.

Există două modalităţi de prelucrare a acestor referinţe. anume:

1. Mulţimea generată a entităţilor discursului poate fi extinsă pentru a include orice acţiune, eveniment, timp, locaţie, sau propoziţie descrisă implicit în clauză;

2. Derivează (atunci când este necesar) referinţele din contextul local.

Ambele abordări sunt la fel de mult folosite în practică.

Prelegerea 15

Propoziţii eliptice şi anaforă

15.1 Referinţe definite bazate pe mulţimi

Să ne ocupăm de situaţia referinţelor definite care folosesc mulţimi sau se referă la mulţimi de obiecte definite anterior.

În general, modificatorii din expresiile substantivale care descriu mulţimi pot defini fie proprietăţi ale mulţimii însăşi, fie proprietăţi ale elementelor din mulţime. Deci expresia substantivală:

Cei trei oameni care au părăsit petrecerea.

descrie o mulţime de cardinalitate trei, în care fiecare membru este o persoană caracterizată prin faptul că a părăsit petrecerea.

Citirea existenţială a acestei *NP* definite va fi ceva de forma:

$\exists! M : |M| = 3 \,\&\, M = \{m | OM(m) \,\&\, PĂRĂSI(m, PETRECERE1)\}$

Unicitatea se bazează pe faptul că exact trei oameni au părăsit petrecerea; altfel, ar fi mai multe seturi posibile care îndeplinesc această proprietate. Pentru comparaţie, descrierea nedefinită

Trei oameni care au părăsit petrecerea.

va fi reprezentată prin

$\exists M : |M| = 3 \,\&\, M \subset \{m | OM(m) \,\&\, PĂRĂSI(m, PETRECERE1)\}$.

Deoarece mulţimea M nu este unică, *petrecerea* poate fi părăsită de mai mult de trei oameni.

Expresiile substantivale la plural folosesc în mod constant referirea indirectă, adesea la submulţimi ale mulţimilor deja introduse.

De exemplu, să considerăm discursul:

1a. Câţiva băieţi şi fete au venit la petrecere.
1b. Băieţii au plecat la miezul nopţii.

Expresia *Băieţii* nu descrie o mulţime prezentă explicit în contextul local ci defineşte o mulţime indirect, anume mulţimea tuturor băieţilor din mulţimea anterior menţionată. Pentru a reţine această interpretare, o translatare a lui 1*b* poate fi:

$PĂRĂSI(B1, 12PM)$ *unde* $B1 = \{x | BĂIAT(x)\} \cap *PROSET*$

în care s-a notat cu $*PROSET*$ o mulţime definită contextual. Dacă notăm cu $S2$ entitatea discursului generată de *Câţiva băieţi şi fete* din $1a$, aceasta se poate rezolva prin

$PĂRĂSI(B1, 12PM)$ *unde* $B1 = \{x|BĂIAT(x)\} \cap S2$.

Ca un alt exemplu, să considerăm discursul

2a. Ion, Anca, Radu şi Maria merg la mare.
2b. Băieţilor le-a fost frică să intre în apă.

În propoziţia $2b$, descrierea *Băieţilor* se referă la mulţimea $\{Ion, Radu\}$, unic definită prin intersecţia mulţimii băieţilor cu $\{Ion1, Anca1, Radu1, Maria1\}$ introdusă de expresia substantivală conjunctivă dată de $2a$.

Această tehnică funcţionează în multe cazuri; probleme apar atunci când mulţimea de referinţă nu este prezentă în universul local al discursului.

Să considerăm discursul:

3a. Paul l-a întâlnit la Mihai pe plajă.
3b. Cei doi băieţi s-au dus apoi la magazin.

Aici $3a$ nu defineşte explicit nici o mulţime. O propunere ar fi ca mulţimea $\{Paul, Mihai\}$ să devină relevantă via un raţionament despre situaţie. În acest caz se poate folosi cunoştinţa generală că atunci când doi oameni se întâlnesc, ei sunt împreună şi deci formează o mulţime nouă.

Dar această tehnică nu funcţionează totdeauna; astfel ea nu se poate aplica pentru discursul:

4a. Paul trăieşte la Bucureşti iar Mihai la Iaşi.
4b. Cei doi nu s-au întâlnit niciodată.

Prelucrarea la modul general a unor asemenea cazuri rămâne problematică.

15.1.1 Referirea la elementele unei mulţimi

Când o mulţime este în contextul discursului, descrierile se pot referi şi la elemente ale sale. Multe exemple folosesc cuvinte specifice care semnalează această comportare.

De exemplu, cuvântul *unul* poate fi semnul unei referiri indirecte la o mulţime; astfel, în discursul:

5a. La turneu, un junior a jucat cu trei mari maeştri.
5b. Cel puţin pe unul l-a învins.
5b'. I-a învins pe toţi înafară de unul.

O reprezentare cât mai completă a lui *unul* poate fi obţinută dacă translatăm această referire în

$$\exists x : x \in *PROSET*$$

unde $*PROSET*$ este determinat referenţial. În $5b$ antecedentul va fi mulţimea marilor maeştri cu care a jucat juniorul în turneu; să o notăm *MAESTRU1*. Atunci forma pentru $5b$ care realizează o referire nedefinită, este:

$\exists x : x \in MAESTRU1 \;.\; ÎNVINGE(x, JUNIOR1)$

Referirea definită realizată în 5*b* este o referire de forma:
$\exists! x : x \in MAESTRU1 \ \& \ \neg ÎNVINGE(x, JUNIOR1)$.

Se pot imagina şi exemple mai complexe, în care nu există nici o modalitate explicită de semnalare când este definită contextual referinţa printr-o mulţime. Astfel:

6a. Silviu a utilizat în drumul lui două mijloace de transport.
6b. Călătoria cu avionul i-a plăcut mai mult.

În acest caz se poate ca în contextul discursului să nu fie nici un obiect cu proprietatea că este avion. Pe de-altă parte, expresia substantivală a selectat un element din mulţimea menţionată anterior şi i-a adăugat proprietăţi noi (care, prezumtiv o determină în mod unic). Deci, presupunând că am notat cu *TRANSP1* entitatea discursului pentru două mijloace de transport, reprezentarea propoziţiei 6*b* este:
$\exists t! : t \in TRANSP1 \ \& \ AVION(t) \ . \ PREFERA(Silviu1, t)$
Deoarece asemenea situaţii conţin proprietăţi necunoscute auditoriului înainte de perceperea propoziţiei, ele sunt adesea dificil de detectat.

15.1.2 Cuantificatori universali pentru mulţimi

Cuantificatorii universali cum ar fi *fiecare* sau *oricare* au o comportare interesantă. Sintactic, ei sunt singulari şi traversează propoziţiile folosind pronume singulare, ca în

Orice om$_i$, când vede marea, aceasta îl$_i$ fascinează.

Între propoziţii, totuşi, aceşti cuantificatori evocă o mulţime ca entitate a discursului; de exemplu:

7a. În parc, fiecare băiat$_i$ a văzut-o pe Sanda.
*7b. * El$_i$ s-a supărat pe ea.*
7b'. Ei$_i$ s-au supărat pe ea.

De remarcat că *El* din 7*b* nu se poate referi la unul din băieţii introduşi în universul cunoaşterii de 7*a*; acest lucru este posibil cu *Ei* din 7*b*'. Aşa că entitatea discursului evocată de 7*a* este mulţimea
$\{x | BĂIAT(x) \ \& \ ÎN(x, PARC1)\}$.

Aceste expresii substantivale folosesc adesea o referinţă definită implicit la o mulţime care limitează domeniul cuantificatorului. De exemplu, entitatea discursului generată pentru *Fiecare fată* din propoziţia

Fiecare fată a văzut-o pe Laura.

luată izolat (fără context) va fi un element din mulţimea $\{x | FATĂ(x)\}$. Aceasta este însă o interpretare foarte dificil de utilizat, deoarece ea exprimă ideea că orice persoană din lume care este fată a văzut-o pe Laura. Să luăm însă discursul:

8a. Mai multe fete s-au dus la petrecere.
8b. Fiecare fată a văzut-o pe Laura.

Când se prelucrează 8*a*, expresia *Mai multe fete* va crea o constantă - să o numim $F1$ - care reprezintă această mulţime de fete. Cu $F1$ folosit drept context local pentru propoziţia 8*b*, interpretarea lui *Fiecare fată* poate fi privit ca un element din $F1$; forma logică (translatarea) lui 8*b* va fi $\forall f : f \in F1 . VEDE1(f, Laura1)$. Această analiză tratează expresia substantivală

"*fiecare fată*" ca o parafrază pentru "*fiecare din fetele*"
şi
"*orice fată*" ca "*oricare din fetele*".

Mulţimea fetelor trebuie determinată referenţial şi este folosită apoi ca domeniu pentru cuantificator.

În alte situaţii, mulţimea este definită folosind referinţa indirectă.

Exemplul 15.1 *Să considerăm 8b în contextul creat de propoziţia*

Mai mulţi băieţi şi fete s-au dus la petrecere.

Aici entitatea $BF1$ a fost creată pentru mulţimea băieţilor şi fetelor, cu proprietatea că s-au dus la petrecere. În acest caz, domeniul este generat printr-o intersecţie, ca la 1b. Translatarea lui 8b va fi în acest caz:

$\exists ! F3 : F3 = \{x | FATĂ(x)\} \cap BF1 \, \forall f : f \in F3 . VEDE1(f, Laura1)$

15.2 Expresii eliptice

După cum s-a evocat în prelegerea anterioară, eliptica apare la folosirea unor clauze care nu sunt propoziţii complete sintactic. Adesea, părţile lipsă pot fi acoperite prin clauzele majore anterioare. Eliptica poate apare ca efect al conjuncţiilor, ca în

9. Nelu a văzut filmul şi Cici a făcut la fel.

unde se înţelege că Cici a văzut şi ea filmul.

Alte cazuri privesc perechi de propoziţii, cum ar fi:

9a. Unii gândesc că Silviu va câştiga cursa de săptămâna viitoare.
9b. Dar n-a reuşit.

O aplicaţie importantă a elipticii se referă la chestionare; de exemplu, între două persoane poate apare următorul dialog:

10a. A: A găsit Mioara banane ?
10b. B: Da.
10c. A: Dar pepene ?

Aici a doua afirmaţie a lui *A* poate fi înţeleasă numai în contextul primei sale întrebări şi este o formă eliptică a lui

A găsit Mioara pepene ?

15.2.1 Restricţii sintactice referitoare la expresiile eliptice

Majoritatea analizelor referitoare la expresiile eliptice se bazează pe ipoteza că structura completă a clauzei eliptice corespunde structurii clauzei precedente. Odată identificată corespondenţa structurală dintre cele două clauze, interpretarea semantică a clauzei anterioare poate fi actualizată cu noua informaţie adusă de clauza eliptică, pentru a produce o nouă interpretare.

Să considerăm din nou propoziţia 9, a cărei analiză este făcută în Figura 15.1:

Figura 15.1:

Clauza contextuală: *Nelu a văzut filmul.*
Structura sintactică:
(*S* (*NP Nelu*)
(*VP a văzut*
(*NP filmul*)))
Forma semantică: *VEDE1(Nelu1, FILM24)*
Clauza eliptică: *Cici a făcut la fel.*
Structura sintactică:
(*S* (*NP Cici*)
(*VP a făcut*))

Comparând structurile sintactice, va fi găsită o similitudine între *Nelu* şi *Cici*. Aceasta va determina modificări cerute de forma semantică, în modul următor:

Întâi *Nelu1* va fi scos din forma semantică abstractă a primei clauze, obţinându-se (în termeni de λ-calcul - pentru detalii, a se vedea [3]):

$\lambda p\, VEDE1(p, FILM24)$.

Această formă este aplicată apoi noii informaţii, anume interpretarea lui *Cici* pentru a produce forma semantică a clauzei eliptice:

$VEDE(Cici1, FILM24)$.

Tehnica exemplificată anterior lucrează şi pentru cazuri mai complexe. Astfel, un exemplu clasic în literatură (numit *problema identităţii colaterale*) foloseşte discursuri voit ambigui, cum ar fi:

11a. Mihai şi-a sărutat soţia.
11b. La fel şi Sandu.

Propoziţia 11*b* poate însemna faptul că *Sandu* a sărutat soţia lui *Mihai* (interpretare numită *citire colaterală*) sau că şi-a sărutat propria sa soţie. Structura sintactică va lucra ca mai sus şi va găsi o corespondenţă între *Mihai* şi *Sandu*.

Să presupunem că forma semantică a lui 11*a* este

SĂRUTA1(Mihai1, SOŢIA_LUI(Mihai1)).

Pentru a genera forma semantică a lui 11*b*, această formă trebuie abstractizată; sunt două feluri de a o abstractiza:

λp *SĂRUTA1(p, SOŢIA_LUI(Mihai1))* şi
λp *SĂRUTA1(p, SOŢIA_LUI(p))*.

Aplicarea lui *Sandu1* în prima formă va conduce la

SĂRUTA1(Sandu1, SOŢIA_LUI(Mihai1))

iar în a doua formă, produce:

SĂRUTA1(Sandu1, SOŢIA_LUI(Sandu1)).

Deci ambele citiri pot fi obţinute formal, similar intuiţiei.

15.2.2 Un algoritm bazat pe sintaxă

Intrare:

- Structura sintactică din contextul local;
- O structură sintactică parţială generată de clauza eliptică.

Algoritm: Constă în găsirea corespondenţei structurale dintre două clauze. Aceste corespondenţe pot fi aflate prin căutarea unui pattern derivat din fragmentul de intrare.

1. Pentru a afla fragmentele care pot fi sintactic paralele în structură, se începe cu un tip care conţine toată informaţia din fragmentul de intrare, exceptând cuvintele aflate în structură (substantive sau adjective). Cuvintele auxiliare (prepoziţii, articole) sunt păstrate.
2. Dacă acest pattern nu conduce la o ieşire corectă prin înlocuirea cu elementele aflate în clauza curentă, se reia încercarea, păstrând mai puţină informaţie în pattern (de exemplu se elimină restricţiile de număr, articolele, prepoziţiile, modificatorii etc).
3. Dacă s-a găsit o corespondenţă, STOP.

   Altfel, se reia pasul anterior.

Observaţii:

- Procesul este mult influenţat de factorii sintactici ai textului.
- Un analizor sintactic bottom-up poate produce această analiză, chiar dacă o analiză completă a propoziţiei nu poate fi realizată.
- Pentru anumite construcţii, cum ar fi VP eliptice, corespondenţa este realizată neambiguu. O astfel de expresie verbală constă din verbul *a face* (sau construcţii echivalente: *a acţiona, a se comporta, ...*) urmat de un modificator (*la fel, tot aşa, identic, din contră, ...*) care indică apariţia unei construcţii verbale eliptice; în acest caz, corespondenţa se face între cei doi subiecţi.
- Pentru alte construcţii, aflarea acestei corespondenţe este mai dificilă.

Exemplul 15.2 *Să reluăm dialogul* 10 *în care NP din clauza eliptică corespunde obiectului NP din clauza precedentă.*

> **Clauza contextuală**: *A găsit Mioara banane ?*
> **Structura sintactică**:
> (*S*[*inter*] (*VP găsit*)
> (*NP*1 *Mioara*
> (*NP*2[3*p*] *banane*)))
> **Forma semantică**: ? *GĂSIT*1(*Mioara*1, *Banane*2)
>
> **Clauza eliptică**: *Dar pepene ?*
> **Structura sintactică**:
> (*S*[*inter*] (*DAR dar*)
> (*NP pepene*))

Aici, patternul iniţial este

(*VP*[3*s*] *NP*1[*nume*] *NP*2[3*s*, *plural*]).

Această structură sintactică a clauzei contextuale este instanţiată cu constituienţii care pot fi înlocuiţi; singurul posibil ar fi "pepene" pe poziţia NP2; dar nu se verifică coincidenţa de număr.

Se reia procedeul cedând informaţie, anume cu patternul

(*VP*[3*s*] *NP*1[*nume*] *NP*2[3*s*]).

Aici se găseşte o înlocuire posibilă a expresiei "banane" cu "pepene" şi se ajunge la construcţia corectă

A găsit Mioara pepene ?

Dacă şi la acest pas nu s-ar fi reuşit o construcţie corectă, se continua cu relaxarea condiţiilor, încercand patternul

(*VP NP*1[*nume*] *NP*2), *apoi*
(*VP NP*1[*nume*]) *urmat de*
(*VP NP*1), *iar în final*
(*VP*).

Formele eliptice care constau din o secvenţă de constituienţi sunt mai complicate. Să considerăm exemplul:

12a. A: A pus tata bananele pe masă ?
12b. B: Da.
12c. A: Şi îngheţata în frigider ?

Interpretatea corectă a lui 12*c* va folosi doi constituienţi independenţi (în loc de unul), anume *NP îngheţata* şi *PP în frigider* (care este un complement al verbului *a pune*). Căutarea completării patternului se face pentru fiecare constituient separat, cu restricţia păstrării ordinii între ei.

15.2.3 Preferinţe semantice

Algoritmul discutat anterior nu identifică totdeauna în mod unic fragmentele corecte.
Să considerăm ca exemplificare dialogul:

A: A pus tata îngheţata în frigider ?
B: Nu.
A: Dar friptura ?

În acest caz, patternul generat de fragmentul de intrare oferă posibilităţi de înlocuire pentru trei *NP*-uri: *tata, îngheţata, frigider*; sintactic, toate pot fi înlocuite cu şanse egale de *friptură*. Este necesară o informaţie semantică pentru a indica preferinţa pentru *friptură*. La fel, dacă replica lui *A* ar fi fost

Dar menajera ?

atunci se înlocuia *tata*, iar pentru replica

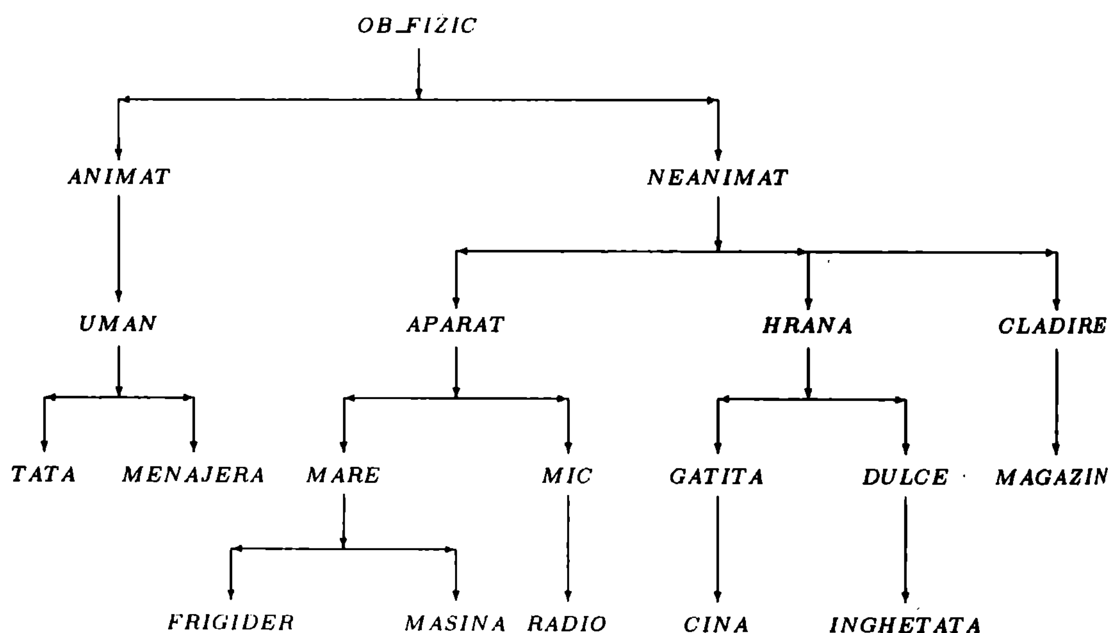
Pe masă ?

se înlocuia *în frigider*.

O tehnică care să ia în considerare aceste informaţii semantice ar fi introducerea unei măsuri de similaritate semantică între fragmentele de intrare şi fragmentele finale, iar apoi algoritmul să preia fragmentul cel mai apropiat conform cu această măsură.

Pentru exemplul dat, să presupunem că sistemul are o taxonomie ierarhică dată în Figura 15.2:

Figura 15.2:



Să luăm cea mai simplă măsură de similaritate, anume distanţa dintre noduri.
Pentru exemplul dat, distanţele vor fi:

$d(cin\breve{a}, tat\breve{a}) = 7$ $\quad d(cin\breve{a}, \hat{\imath}ngheţat\breve{a}) = 4$ $\quad d(cin\breve{a}, frigider) = 6.$

Dacă se alege pentru înlocuire constituientul a cărui familie este cea mai apropiată de *cină*, acesta este *îngheţată*.

Evident, pot apare şi cazuri în care această construcţie nu funcţionează cu rezultate bune. Dacă interpretorul semantic nu este capabil să analizeze corect structura rezultată, pot fi obţinute alte alegeri.

Să considerăm următorul dialog:

A: Ai văzut in magazin pe menajeră ?
B: Da.
A: Dar frigiderul ?

Aici, o înlocuire conform algoritmului anterior va conduce la

Ai văzut în frigider pe menajeră ?

deoarece *d(frigider, menajeră)* = 7, *d(frigider, magazin)* = 5.

De obicei însă, regulile semantice sunt suficient de bogate pentru a determina improbabilitatea unei asemenea afirmaţii. Algoritmul va sugera atunci alternativa următoare, cu *menajeră* ca posibil element înlocuit, ceea ce duce la o construcţie semantic corectă:

Ai văzut în magazin frigiderul ?

15.3 Anafora de suprafaţă

Există o clasă de expresii referenţiale care introduc obiecte noi, legate de obiectele din universul discursului. Aceste cazuri sunt numite *anafore*[1] *de suprafaţă* deoarece multe teorii se referă la ele în termenii formei de suprafaţă a propoziţiei precedente, spre deosebire de cazurile tratate anterior care se refereau la obiectele din contextul discursului şi sunt numite *anafore de adâncime*.

Să considerăm câteva exemple de anaforă de suprafaţă:

*13a. Spune-mi gradul lui Mateescu în CSC*271.
*13b. Dă-mi-l şi în MTH*444.
*13b'. Spune-l pe al lui Ionescu, tot în MTH*444.

Obiectele la care se face referinţă în 13*b* şi 13*b*′ nu sunt menţionate în 13*a*. Dar contextul creat de 13*a* sugerează să interpretăm fiecare termen ca un grad diferit (gradul lui *Mateescu* şi respectiv al lui *Ionescu* în *MTH*444).

Există componente sintactice specifice care să semnaleze anafora de suprafaţă. O clasă largă foloseşte expresiile substantivale cu un cuvânt lipsă:

- *lui Mateescu în MTH*444.

Se pot utiliza construcţii adverbiale, de tipul: *ca, la fel cu, şi el*, care indică explicit o comparaţie cu ceva din contextul discursului.

Există mai multe metode propuse pentru a lucra cu anafora de suprafaţă.

- Una din ele foloseşte identificarea pe o structură sintactică/semantică, aşa cum s-a procedat la construcţiile eliptice. Ideea se bazează pe faptul că expresia anaforei de suprafaţă este un constituient incomplet. Expresia este procesată

[1] Anaforă: procedeu stilistic constând în repetarea unui cuvânt la începutul mai multor fraze sau părţi de frază pentru accentuarea unei idei sau pentru obţinerea unor simetrii.

parcurgând arborele sintactic al propoziţiei precedente, până se află un constituient a cărui structură corespunde cu structura căutată, apoi se extrage informaţia respectivă - pentru ca împreună cu informaţia curentă să formeze o expresie nouă care poate fi analizată cu procedeele obişnuite şi să producă o interpretare.

În exemplul de sus, pentru $13b'$ se procedează în acest fel până se selectează constituientul *gradul*, care va înlocui pronumele *-l*.

- O altă abordare se bazează pe semantică. Ea presupune că informaţia care lipseşte este caracterizată de o anumită mulţime anterior menţionată. Pentru aceasta se prelucrează abstracţiile mulţimilor definite în contextele anterioare.

  Astfel, propoziţia $13a$ va evoca mulţimea (cu un singur element) a gradelor lui *Mateescu*:

  $$\{g|\ GRAD(g,\ Mateescu,\ CSC271)\}.$$

  O abstractizare a acesteia poate fi mulţimea gradelor oricui

  $$\{g|\exists p:\ PERSOAN\breve{A}(p)\ .\ GRAD(g,p,CSC271)\}$$

  sau mulţimea gradelor lui *Mateescu* pe orice scală:

  $$\{g|\exists c:\ SCAL\breve{A}(c)\ .\ GRAD(g,\ Mateescu,\ c)\}.$$

  Anafora de suprafaţă va selecta una din aceste mulţimi şi îi va adăuga proprietăţi suplimentare, bazate pe informaţia nou introdusă în frază.

Modul de inferenţă necesită totuşi un algoritm care să ştie să incorporeze modificatorii adiţionali. Aceştia sunt destul de dificili şi nu reuşesc să acopere întotdeauna toate cazurile.

Prelegerea 16

Agent conversaţional

16.1 Ce este un agent conversaţional

În această secţiune, prin **agent conversaţional** vom înţelege o fiinţă (sau ceva similar) care poate participa la un dialog purtat în limbaj natural. Aplicaţiile unei astfel de construcţii sunt numeroase, plecând de la tipul de comunicare *om - maşină* care se doreşte a se realiza în viitor, până la stabilirea subtilităţilor caracteristice unei limbi.

Primele întrebări care se pun relativ la un agent conversaţional sunt:
De ce trebuie el să vorbească ?
Ce motiv are să spună orice sau să încerce să înţeleagă ce i se spune ?

De exemplu, să considerăm o bază de date de tip întrebare - răspuns realizată printr-un program, care pentru fiecare intrare execută următoarele operaţii:

1. Analizează şi interpretează întrebarea sub o formă logică reprezentând o cerinţă;

2. Execută cerinţa folosind baza de date şi generează o ieşire.

Poate fi considerată aceasta un agent conversaţional ?

Răspunsul este negativ, deoarece ea nu va putea reacţiona pozitiv la nici o cerinţă care nu satisface regulile - foarte precise - de intrare; nici o altă formulare a întrebării, oricât de apropiată de cea folosită de program, nu este acceptată. În plus, ea nu este capabilă de nici o comportare independentă care să poată primi calificativul de *inteligentă.*

Ce trăsături trebuie să aibă însă un agent pentru a putea fi calificat cu *"inteligenţă"* ? Credem că cele mai specifice sunt:

- *Percepţii*: agentul trebuie să fie capabil să perceapă lumea din jurul său;

- *Cunoştinţe*: agentul trebuie să aibă o reprezentare conştientă a stării universului în care este încorporat[1];

[1] prin această definire se va înţelege "ceea ce crede agentul că este adevărat"

- *Dorinţe*: agentul trebuie să aibă răspunsuri (pozitive sau negative) la diverse stări ale universului, dispunând de un mod de a le compara;

- *Planificare/raţionament*: trebuie să fie capabil să raţioneze asupra căilor de a atinge alte stări;

- *Strategii*: agentul trebuie să aibă capacitatea de a decide acţiuni pentru schimbarea stării universului;

- *Intenţii*: trebuie să poată menţine strategia acţiunii decise;

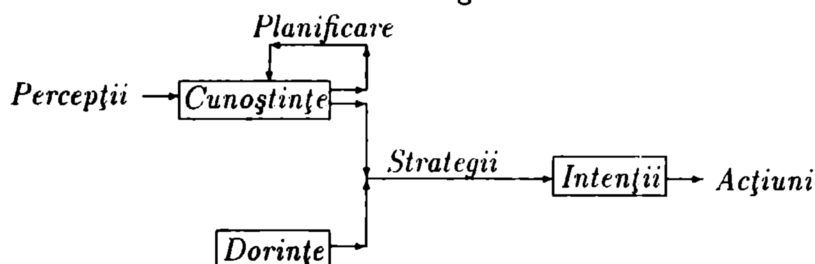
- *Acţiuni*: agentul trebuie să fie capabil să acţioneze şi deci să-şi schimbe starea.

Un agent conversaţional "trăieşte" într-un univers al limbajului şi logicii. Singurele sale percepţii sunt afirmaţiile care i se adresează, iar singurele acţiuni sunt afirmaţiile pe care le generează.

Din cele şapte aspecte ale comportării inteligente menţionate anterior, patru - ***percepţii, planificări, strategii, acţiuni*** sunt procese, iar celelalte trei - ***cunoştinţe, dorinţe, intenţii***, sunt componente ale stării cognitive ale agentului.

Un model pentru un agent inteligent este dat în Figura 16.1

Figura 16.1:



Se observă de aici că agentul îşi actualizează permanent *cunoştinţele* bazate pe *percepţii*, foloseşte aceste cunoştinţe pentru a raţiona relativ la posibile *strategii* pentru a-şi *planifica* anumite *intenţii* bazate pe cunoştinţele şi *dorinţele* sale, intenţii pe care le realizează prin *acţiuni*.

Pentru a aplica un astfel de model la limbaje, trebuie precizată noţiunea de *afirmaţie*.

Definiţia 16.1 *În limbaj natural, o afirmaţie este* **un act de vorbire**; *prezenţa ei este fundamentală în dezvoltarea agentului conversaţional.*

Actele de vorbire sunt precizate prin anumite verbe, cum ar fi *a întreba, a cere, a informa, a nega, a mulţumi, a felicita, a confirma* etc.

16.1.1 Limbajul ca activitate multi-agent

Limbajul foloseşte totdeauna mai mulţi agenţi. Alte forme de comunicare - cum ar fi *scrisul* - fac doar o extensie în timp a comunicării. Orice persoană comunică cu

scopul final de a afecta starea cognitivă a celorlalţi. Monologul **nu** constituie un agent conversaţional.

În plus, comunicarea nu poate apare dacă unul din agenţi nu percepe încercarea celuilalt de a comunica.

Exemplul 16.1 *Să presupunem că Ion şi Ana sunt amanţi şi pun la punct o schemă în care Ion o poate vizita pe Ana numai dacă aceasta lasă fereastra deschisă. Astfel, fereastra constituie un mijloc de comunicare între cei doi.*

Într-o zi, fiind foarte cald, Ana deschide fereastra pentru a se răcori; Ion intră fiind sigur că acesta este semnalul convenit. Nu mai spunem ce se întâmplă.

Altă dată, deşi este cald, Ana deschide fereastra pentru a-l anunţa pe Ion că poate veni. Dar acesta nu intră, convins că deschiderea ferestrei s-a făcut doar pentru a răcori camera.

Ambele cazuri sunt situaţii în care fiecare din parteneri nu recunoaşte intenţia de comunicare a celuilalt.

Deci, o comunicare poate apare numai atunci când un agent intenţionează să transmită ceva, iar celălalt agent recunoaşte această intenţie.

O altă cerinţă pentru comunicare este existenţa unui set de convenţii cunoscut de toţi agenţii, cu semnificaţii neambigui.

În exemplul de sus există numai două simboluri convenite: fereastra deschisă (semn de siguranţă) şi fereastra închisă (semn de pericol). Astfel, *Ion* şi *Ana* şi-au definit propriul lor limbaj, care funcţionează perfect - din păcate, numai când nu este prea cald.

Conform cu Austin (1962), atunci când se spune ceva, se realizează trei acţiuni:

- **actul locuţionar**: realizarea unei afirmaţii ca o secvenţă de cuvinte;
- **actul ilocuţionar**: acţiunea prin care vorbitorul realizează actul locuţionar;
- **actul perlocuţionar**: acţiunea care apare ca urmare a afirmaţiei.

Revenind la Exemplul 16.1: acţiunea de deschidere a ferestrei reprezintă actul locuţional; el poate fi realizat cu sau fără intenţia de a comunica. Când *Ana* deschide fereastra cu intenţia de a semnala că drumul este liber, ea execută un act ilocuţional. Dacă *Ion* vede fereastra deshisă şi înţelege semnificaţia mesajului - atunci actul perlocuţional - convingerea că este aşteptat - este realizat.

Unele verbe se folosesc pentru a descrie acte ilocuţionale, iar altele - pentru acte perlocuţionale. Actele ilocuţionale pot fi folosite în propoziţii care exprima explicit acţiunea ce trebuie realizată; aceste utilizări se numesc *explicit performative*.

De exemplu:

> *Promit că mâine nu mai pierd vremea.*
> *Prin aceasta vă informez că aveţi de plătit o amendă.*

Pe de-altă parte, verbul *a convinge* descrie un act perlocuţional. De remarcat că o intenţie poate fi transmisă cuiva, dar apariţia ei este sub controlul agentului are face afirmaţia. Deci, nu se poate spune:

** Prin aceasta vă conving că aveţi de plătit o amendă.*

Un act perlocuţional poate fi realizat fără ca celălalt agent să fie nevoit să recunoască intenţia de comunicare.

De exemplu, *Paul* îl poate convinge pe *Dorin* că are de plătit o amendă lăsând actul respectiv în cutia poştală.

O problemă dificilă în înţelegerea limbajului constă în identificarea actului ilocuţional creat de vorbitor. Astfel, acelaşi act locuţionar

Ştiţi cât este ceasul ?

poate genera trei acţiuni ilocuţionale diferite: o întrebare cu răspuns *Da/Nu*, o întrebare despre ora exactă, sau o întrebare despre preţul unui obiect care se vinde.

Identificarea sensului corect al unui act locuţional este problema fundamentală din teoria vorbirii.

16.2 Reprezentarea cunoştinţelor

Sunt multe moduri de reprezentare a cunoştinţelor unui agent, în funcţie de cât de profunde sunt ele şi cât de dornic de comunicare este agentul.

Astfel, să folosim ca bază de cunoştinţe KB toate evenimentele acestui an şi să spunem doar că tot ce este în KB este cunoscut de agent[2]. Această definiţie va produce un agent limitat. El nu va putea - de exemplu - să-şi reprezinte informaţia despre cunoştinţele celorlalţi agenţi, astfel că nu este motivat să comunice. Mai mult, nu se poate proteja, deoarece nu poate face deosebire între ceea ce ştie şi ce observă adevărat în univers - deci percepţia nu îi foloseşte pentru a afla ceva nou.

Cea mai simplă extensie este divizarea KB-ului în zone numite *spaţii de cunoştinţe*. Un astfel de spaţiu este un set de propoziţii care reprezintă cunoştinţele unui anumit agent. Practic, va fi necesar un predicat modal - *Cred* - care leagă un agent de un spaţiu de cunoştinţe, şi ia valoarea "adevărat" dacă şi numai dacă spaţiul respectiv caracterizează cunoştinţele agentului.

Faptul că o propoziţie p este în spaţiul de cunoştinţe al agentului A va fi înţeleasă prin afirmaţia:

$$A \text{ crede că } p.$$

Pentru că unele propoziţii folosind predicatul *Cred* pot apare şi în alte spaţii, va fi necesară o ierarhizare a spaţiilor de cunoştinţe.

Exemplul 16.2 *Fie un KB descris în Figura 16.2. Cele două coloane reprezintă acelaşi set de propoziţii; coloana din stânga le listează în logica modală(cu predicatul "Cred"), iar cea din dreapta le grupează pe spaţii de cunoştinţe.*

Acest KB strânge cunoştinţele lui Ion în spaţiul BS1 - deci Ion crede că "Dick este câine", "câinii muşcă", şi "cunoştinţele lui Ana sunt strânse în BS2".

Spaţiul BS2 descrie ceea ce crede Ion că crede Ana: "Dick este un câine", "câinii muşcă", şi că "orice câine care muşcă poartă zgardă".

[2]O propoziţie este în baza de cunoştinţe a unui agent dacă agentul crede că ea este adevărată.

Figura 16.2:

Cred(Ion1,Câine(Dick1)) *Cred(Ion1,$\forall x$: Câine(x).Muşca(x))* *Cred(Ion1, CRED(Ana1,Câine(Dick1)))* *Cred(Ion1, CRED(Ana1,* *$\forall x$: Câine(x).Muşca(x)))* *Cred(Ion1, CRED(Ana1,* *$\forall x$: Câine(x).Muşca(x)$\supset$ Zgardă(x)*	*CRED(Ion1,BS1)* **BS1:Cunoştinţele lui Ion** *Câine(Dick1)* *$\forall x$: Câine(x).Muşca(x)* *CRED(Ana1, BS2)* **BS2: Cunoştinţele lui Ion despre cunoştinţele lui Ana** *Câine(Dick1)* *$\forall x$: Câine(x).Muşca(x)* *$\forall x$: Câine(x).Muşca(x)$\supset$ Zgardă(x)*

Într-un astfel de model, inferenţa este dată relativ la un spaţiu. De exemplu, în raport cu $BS2$, putem trage concluzia că "Dick poartă zgardă", deci "Ion crede că Ana crede căDick poartă zgardă".

O astfel de concluzie nu poate fi trasă în spaţiul $BS1$: "Ion nu crede că Dick poartă zgardă".

Definiţia 16.2 *Propoziţiile listate în spaţiul de cunoştinţe ale unui agent formează cunoştinţele* **explicite** *ale agentului.*

Propoziţiile care pot fi derivate prin inferenţe formează cunoştinţele **implicite** *ale agentului.*

Există o distincţie netă între ce crede un agent şi ce crede acel agent că alţi agenţi cred. Aceasta se reliefează clar când asistăm la o conversaţie între două persoane cu vederi diferite.

De exemplu, să presupunem că Ana crede că o carte *Zece negri mititei*, este pe biroul lui Ion, dar crede că Ion crede că această carte este în bibliotecă. Dacă Ana întreabă:

Ai citit cartea de pe birou ?

ea se va referi la *Zece negri mititei*, indiferent ce va spune Ion despre carte.

Deci, cunoştinţele unui agent relativ la cunoştinţele altui agent joacă un rol important în înţelegerea oricărui limbaj.

Problema care apare este cât de adânc trebuie iterat nivelul de cunoştinţe pentru a înţelege toate limbajele.

Exemplul 16.3 *Să presupunem că Ion crede că Ana l-a minţit când i-a spus că plouă. În acest caz, Ion va crede următoarele:*

Nu plouă.
Ana crede că nu plouă.
Ana crede că Ion crede că plouă.

Amândoi cred că nu plouă, dar Ion mai crede că Ana crede că el se gândeşte că plouă, pentru că el crede că ea crede că minciuna ei a fost crezută.

În exemplul de sus a fost introdus un al treilea nivel; evident că se pot introduce similar un număr arbitrar de nivele de cunoştinţe. Un mod de soluţionare a problemei constă în introducerea noţiunii de ***cunoştinţe comune*** - cunoştinţele pe care ambii agenţi le cunosc şi fiecare crede că celălalt le cunoaşte. Acestea pot constitui elementul de început al oricărei conversaţii.

16.2.1 Despre cunoştinţele altora

O altă problemă extrem de dificilă este cuantificarea cunoştinţelor pe care agentul A crede că le posedă agentul B, dar pe care A nu le ştie.

De exemplu, *Ion* crede că *Ana* ştie numele mamei ei, dar el nu cunoaşte acest nume. Folosind logica modală, se introduce în spaţiul de cunoştinţe al lui *Ion* propoziţia

1. $\exists\,!x$. *Cred(Ana1,Nume(Mama(Ana1),x))*

De remarcat că o afirmaţie de forma:

2. Cred(Ana1, $\exists\,!x$. *Nume(Mama(Ana1),x))*

semnifică cu totul altceva: *Ana ştie că mama ei are un nume !*

Se introduce un operator nou *StieDespre* care formalizează faptul că un agent cunoaşte un obiect unic care satisface o anumită descriere. Folosind notaţii de λ-calcul ([3]), el se poate defini astfel:

$$StieDespre(A, \lambda x, P_x) \equiv \exists!\, y\ .\ Cred(A, ((\lambda x\ P_x), y)) \equiv \exists y\ .\ Cred(A, P_y)$$

Cu această notaţie, formula 1 se scrie
StieDespre(Ana1,λx *Nume(Mama(Ana1),x))*

O altă formă de cunoştinţe reprezintă ceea ce un agent A ştie că alt agent crede că un anumit fapt este adevărat (fără ca A să aibă idee dacă acesta este adevărat sau nu).

De exemplu, *Ion* poate crede că *Ana* ştie dacă a luat examenul sau nu. Acest lucru poate fi reprezentat în logica modală ca o disjuncţie de cunoştinţe. Problema este unde va fi scrisă această disjuncţie: înafara sau în interiorul operatorului modal *Cred*.

Astfel, dacă disjuncţia este interioară operatorului, adică:

Cred(Ana1, Luat(Ana1, Examen)$\vee\neg$ *Luat(Ana1,Examen))*

aceasta reprezentă faptul că *Ana* ştie că a luat examenul sau ştie că nu l-a luat. Este o informaţie ştiută sigur de *Ana* sau oricare alt agent raţional (conform principiului terţului exclus).

Celălalt caz (cu disjuncţia în exteriorul operatorului) se reprezintă:

Cred(Ana1, Luat(Ana1, Examen))$\vee\neg$ *Cred(Ana1, Luat(Ana1,Examen))*

Aici, *Ana* ştie că a luat examenul sau nu ştie că l-a luat.

Pentru reprezentarea în spaţiul cunoştinţelor, se foloseşte un operator nou *StieDacă*, definit prin

$$StieDac\breve{a}(A, P) \equiv Cred(A, P) \vee Cred(A, \neg P)$$

16.3 Dorinţe, Intenţii şi Planuri

La prima lectură, se pare că cele două noţiuni *Dorinţă* şi *Intenţie* au aceeaşi semnificaţie: *scop, ţintă, vrere.* Să încercăm să punem în evidenţă diferenţele dintre ele, justificând printre altele şi realizarea de stări cognitive distincte.

În general, *dorinţele* reflectă stările universului pe care un agent le găseşte plăcute sau neplăcute. Un agent poate avea mai multe dorinţe, adesea aflate în conflict unele cu altele.

De exemplu, vrei să ai câteva zile libere şi să mergi la munte. Pe de-altă parte, ai un examen şi doreşti să iei o notă bună, deci vei sta acasă pentru a învăţa. Aceste două dorinţe intră în conflict şi comportarea va rezulta printr-un compromis care să împace *şi capra şi varza.*

Intenţiile însă nu vor fi niciodată în conflict. Nu poţi niciodată să intenţionezi să mergi la munte pentru recreere şi să stai să înveţi. Intenţiile sunt legate puternic de comportare şi nu sunt posibile două acţiuni în acelaşi timp.

Există două noţiuni ale termenului *intenţie.*

Prima - *intenţia acţiunii* - se referă la proprietatea unei acţiuni intenţionate.

De exemplu, se dă o mare importanţă diferenţei între o alergare fără ţintă, făcută numai de dragul de a alerga, şi o alergare efectuată pentru a ajunge într-un loc precis.

A doua este numită *direcţia viitoarei intenţii*, şi reflectă deciziile care vor fi luate pentru viitoarea acţiune.

De exemplu, dacă după o anumită deliberare ai decis că este mai important să laşi de-o parte pregătirea pentru examen şi să mergi la munte, asta înseamnă că ai adoptat o direcţie a viitoarei decizii: destinderea la munte.

Deşi pe baza unei direcţii a viitoarei intenţii se ajunge la o acţiune intenţionată, nu întotdeauna o astfel de intenţie este şi realizată. De exemplu, după ce ai decis că mergi la munte, descoperi că nu ai cu cine, că toţi prietenii învaţă, te apucă un val de remuşcări şi revii să înveţi. Nu este bineînţeles un scenariu agreabil, dar este pe deplin posibil.

În problemele de reprezentare a cunoştinţelor, acest concept de direcţie a viitoarei acţiuni este cel mai sensibil. Vom introduce un operator numit *Intenţie* care leagă un agent de acţiunea pe care intenţionează să o întreprindă. Există mai multe restricţii care vor fi luate în considerare:

Restricţia de raţionalitate 1: Un agent nu poate intenţiona să facă două acţiuni care se exclud reciproc.
Restricţia de raţionalitate 2: Un agent nu poate intenţiona să pornească o acţiune pe care nu o poate realiza.

Aceste două restricţii sunt greu de exprimat în formă logică. În cele ce urmează vom considera numai acţiuni care satisfac cele două restricţii.

Evident, planurile, scopurile și planificarea sunt legate de intenții. Dar legătura este mai complicată decât la prima vedere.

De exemplu, propoziția

Ion plănuiește jefuirea unei bănci.

poate avea două semnificații diferite:

- *Ion* intenționează să atace o bancă;
- *Ion* a lucrat la o schemă cu care cineva poate jefui o bancă.

În prima descriere, a avea un plan semnifică una din direcțiile viitoarelor intenții, în timp ce în a doua descriere, a avea un plan înseamnă în principiu că *Ion* știe cum să jefuiască o bancă. Aici deci, *un plan* reprezintă un set de acțiuni care conduc la un anumit efect (ca o rețetă). A avea o rețetă nu înseamnă neapărat că există o intenție.

În general, noțiunea *plan* este folosită în ideea de *rețetă.*

Definiția 16.3 *Un plan este un set de acțiuni cu un anumit efect, numit scopul planului.*

Un scop este o propoziție care reprezintă o rezultantă a acțiunilor descrise de plan, sau chiar o acțiune realizată de acțiunile planului.

Recunoașterea unui plan se referă la problema construirii unui plan - rețetă, format dintr-un set de acțiuni.

Pentru a evita confuzii, nu vom folosi termenul de *plan* legat de noțiunea de *intenție.*

Structura intențională a unui agent este o componentă mintală a sa (cum sunt *cunoștințele* și *dorințele*). Ea apare în momentul când agentul începe realizarea unui plan. Există o corespondență strânsă între structura unui plan și structura intențională rezultată.

Exemplul 16.4 *Să presupunem că Ion cunoaște un plan foarte simplu pentru a merge la munte; stând lângă gară, va lua trenul. Planul este o secvență de cunoștințe care spun că dacă Ion va lua trenul în direcția corectă, el va ajunge la munte. La sfârșit de săptămână, Ion vrea să meargă la munte. Cunoscând acest plan, el îl poate adopta. Își va pune în aplicare intenția luând trenul spre munte.*

În Figura 16.3 sunt reprezentate cunoștințele lui Ion despre plan și structura intențională pentru adoptarea lui.

Figura 16.3:

$Cred(Ion1,$ $Genera(IaTren(Ion1),$ $MergeMunte(Ion1)))$	$Intentie(Ion1, IaTren(Ion1))\&$ $Intentie(Ion1, MergeMunte(Ion1))\&$ $Intentie(Ion1, Genera(IaTren(Ion1),$ $MergeMunte(Ion1)))$
Știind un plan de mers la munte	**Intenția de a merge la munte**

De remarcat că intențiile trebuie să includă atât acțiunile cât și relațiile dintre acțiuni. De exemplu:

- *Dacă se omite acţiunea "IaTren", atunci Ion va dori să meargă la munte dar nu va avea un plan cum să ajungă acolo.*
- *Dacă se omite acţiunea "MergeMunte", Ion va intenţiona să ia trenul, dar nu necesar pentru a merge la munte.*
- *Dacă se omite relaţia "Genera", atunci Ion poate efectua independent fiecare acţiune (poate merge cu trenul, apoi poate să ajungă la munte cu o maşină).*

Este interesant faptul că intenţiile se pot exprima în funcţie de acţiuni. Pollack (1990) a introdus un operator *Cu* astfel:

> *Fiind date acţiunile* α, β, $Cu(\alpha, \beta)$ *este acţiunea de efectuare a lui* α *făcând* β.
> Atunci, a treia intenţie din Figura 16.3 va fi
> *Intentie(Ion1,Cu(MergeMunte(Ion1), IaTren(Ion1)))*

16.4 Actele de vorbire ca acte comunicative

După cum am văzut, actele de vorbire sunt acţiuni realizate prin procesul emiterii de sunete. Ele vizează doar două tipuri de acte: cele ilocuţionare şi perlocuţionare.

Din puncte de vedere ilocuţionar, se pare că sunt cinci tipuri (clase) de acte de vorbire. Fiecare acţiune din clasă este specificată ulterior prin condiţii suplimentare.

Cele cinci clase sunt:

1. **Clasa reprezentativă**: vorbirea este orientată pentru a sublinia adevărul afirmaţiilor prezentate. Aici intră acte descrise de verbe ca *a informa, a nega, a afirma, a confirma*;

2. **Clasa directivă**: vorbirea încearcă să influenţeze intenţiile şi comportarea altui agent. Sunt incluse acţiunile descrise de verbe ca *a cere, a comanda, a ruga, a invita, a întreba*;

3. **Clasa orientativă**: vorbitorul se referă la acţiuni viitoare, cum ar fi cele descrise de *a promite, a comite*;

4. **Clasa expresivă**: vorbitorul exprimă o stare psihologică sau o reacţie; aici intră verbe ca *a felicita, a mulţumi, a slăvi, a mulţumi, a ura*;

5. **Clasa declarativă**: vorbitorul realizează anumite convenţii sau acţiuni rituale. Exemple caracteristice sunt *a da foc, a se ruga, a se resemna.*

Se pare însă că datorită restricţiilor modelului impuse de planificare, actele perlocuţionare capătă o importanţă deosebită; motivul este că acestea prind efectele intenţiilor pe care le exprimă agenţii prin vorbire.

Exemplul 16.5 *Un agent vrea să construiască un plan prin care să convingă un alt agent că un anumit fapt este adevărat, sau ca celălalt agent să realizeze o anumită acţiune. Pentru aceasta, agentul va construi un act ilocuţionar (din clasa reprezentativă sau din cea directivă) şi speră că celălalt agent să reacţioneze după aşteptări,*

finalizând actul perlocuţionar. Evident, pentru construirea actului ilocuţionar, agentul va iniţia un act locuţionar care va fi folosit la momentul potrivit prin exprimarea unei propoziţii.

Modelul dezvoltat aici foloseşte *actele comunicative*, care corespund actelor perlocuţionare generate de actele ilocuţionare.

De exemplu, actul comunicativ *ConvingePrinInformare* convinge alt agent să creadă anumite propoziţii, spunând că sunt adevărate.

Figura 16.4: Definirea a două acte comunicative

Clasa de acţiune	:	**ConvingePrinInformare**
Roluri	:	*Vorbitor, Auditor, Prop*
Restricţii	:	*Agent(Vorbitor), Agent(Auditor), Propoziţie(Prop), Cred(Vorbitor, Prop)*
Precondiţii	:	*La(Vorbitor), Loc(Auditor))*
Efecte	:	*Cred(Auditor), Prop)*
—		
Clasa de acţiune	:	**MotivatDeCerer(e/i)**
Roluri	:	*Vorbitor, Auditor, Act*
Restricţii	:	*Agent(Vorbitor), Agent(Auditor), Acţiune(Act)*
Precondiţii	:	*La(Vorbitor, Loc(Auditor))*
Efecte	:	*Intenţie(Auditor, Act)*

Exemplul 16.6 *Să considerăm ce trebuie să fie adevărat pentru Andrei pentru a ConvingePrinInformare pe Dana că plouă la mare:*

1. *Andrei spune cuvintele "Plouă la mare" cu intenţia de a o informa pe Dana de acest fapt.*

2. *Dana aude propoziţia "Plouă la mare" şi recunoaşte intenţia lui Andrei de a o informa de acest fapt. De aceea, ea decide că Andrei intenţionează să o facă să creadă că plouă la mare.*

3. *Dana consideră evident faptul că plouă la mare, inclusiv faptul că Andrei spune că plouă.*

4. *Dana ştie că plouă la mare.*

Deşi unele acţiuni par redondante, toţi paşii descrişi mai sus sunt esenţiali. Astfel:
- Dacă lipseşte pasul 1, *Andrei nu spune nimic; astfel s-a eliminat actul locuţional şi - fără condiţia de intenţie - apare un scenariu prin care Andrei scoate nişte sunete oarecare (eventual sub hipnoză). Chiar dacă prin absurd Dana are o revelaţie şi realizează restul paşilor, Andrei poate nega mai târziu că a transmis informaţia că plouă la mare.*

- Prin lipsa pasului 2, *Dana nu aude propoziţia; deci ea aude eventual ceva, dar nu ştie ce, aşa că nu poate comite actul ilocuţional. Chiar dacă va parcurge restul*

paşilor şi va şti în final că plouă la mare, Dana va nega că ştie acest lucru de la Andrei.

- Eliminând pasul 3, Dana - recunoscând că Andrei încearcă să îi transmită o informaţie - va ignora complet această informaţie. Este un gest nepoliticos, dar - chiar dacă în final Dana va şti că plouă la mare, aceasta nu are nici o legătură cu ce a spus Andrei.

- Dacă se sare peste pasul 4, Andrei i-a spus Danei că plouă la mare, dar nu a convins-o. În acest fel, actul "Inform" s-a realizat dar "ConvingePrinInformare" a eşuat.

Acţiunile comunicative asigură legătura principală între raţionamentele agenţilor. În Figura 16.4 sunt reprezentate definiţiile a două acte comunicative de bază în orice model de comunicare: informare simplă (clasa reprezentativă) şi cerere (clasa directivă).

Aceste acţiuni sunt construite în ipoteza că agenţii sunt sinceri. De exemplu, o restricţie a actului *ConvingePrin Informare* este aceea că vorbitorul crede în propoziţia pe care o enunţă.

16.5 Actul comunicativ şi recunoaşterea intenţiilor

Fără abilitatea de a recunoaşte intenţiile agentului partener, acţiunile unui agent vor fi extrem de dificile. În particular, va fi imposibil de înţeles şi acţionat într-un dialog. De aceea va trebui să luăm puţin în considerare această problemă de recunoaştere a intenţiilor şi cum leagă ea vorbirea de actul de interpretare.

În dialog există totdeauna un agent care realizează acţiunea (agent activ) şi un agent care observă acţiunea (agent observator). Ceea ce vom discuta aici este modul în care agentul observator recunoaşte intenţiile agentului activ.

În primul rând, faptul că s-a identificat un plan care include toate acţiunile observate nu înseamnă că acel plan descrie intenţiile agentului. Pot fi mai multe planuri posibile care conţin acţiunile observate, iar agentul activ realizează doar unul din ele. Determinarea exactă a planului corect pare imposibilă teoretic.

După cum s-a menţionat anterior, planul recunoscut trebuie să fie plauzibil, deşi din punctul de vedere al agentului observator poate părea ineficient sau greşit. Practic, aceasta înseamnă că recunoaşterea unui plan se bazează numai pe cunoştinţele comune, la care se pot adăuga eventual unele informaţii (dacă se cunosc) despre cunoştinţele celuilalt agent.

Exemplul 16.7 *Să presupunem că Radu are o cameră a cărei uşă se deschide numai pe baza amprentelor mâinii sale; Sandu crede că la camera lui Radu este o broască simplă, cu cheie. Cunoştinţele lor comune cuprind informaţia că uşa trebuie descuiată pentru a intra în cameră, informaţii despre efecte ale diverselor acte comunicative şi despre situaţia curentă. Amândoi ştiu - de exemplu - că pentru a intra în camera lui Radu, Sandu trebuie să îi ceară permisiunea lui Radu, şi că acesta i-o va da (în mod normal). În aceste condiţii, ce va înţelege Radu din întrebarea lui Sandu:*

Poţi să-mi dai cheia de la camera ta ?

Sărind peste detalii (cum identifică Radu acest act locuţionar ca o cerere pentru cheie), el poate infera un plan plauzibil în jurul acestei acţiuni, astfel:

Afirmaţia este identificată ca o componentă a acţiunii:

MotivatDeCerere(Sandu1, Radu1, Da(Radu1, Sandu1, Cheie(Broasca1)))

Folosind cunoştinţele comune despre actele de comunicare, Radu ştie că Andrei crede că un efect al acestui act este

Intenţie(Radu1, Da(Radu1, Andrei1, Cheie(Brioasca1)))

care la rândul ei, folosind informaţia comună despre intenţie, va motiva acţiunea

Da(Radu1, Andrei1, Cheie(Broasca1))

Cunoştinţele comune despre relaţiile dintre Radu şi Sandu conduc la concluzia

Avea(Sandu1, Cheie(Broasca1))

Aici, Radu crede că Sandu - crezând că la uşă este o broască cu cheie - va realiza acţiunea

Deschide(Sandu1, Broasca1)

Radu concluzionează că această acţiune este intenţia lui Sandu când a cerut cheia. În ideea că acesta pare singurul plan plauzibil, Radu presupune că a definit structura intenţională a lui Sandu. Intenţiile primare pe care le atribuie Radu lui Sandu în acest plan sunt:

Intenţie(Sandu1, MotivatDeCerere(Sandu1, Radu1,
Da(Radu1, Sandu1, Cheie(Broasca1))))
Intenţie(Sandu1, Realiza(Avea(Sandu1, Cheie(Broasca1))))
Intenţie(Sandu1, Deschide(Sandu1, Broasca1))

În practică un sistem computaţional care foloseşte un astfel de algoritm va conduce la o listă de intenţii posibile, care va servi la ordonarea orizontului de aşteptări. Cât de departe trebuie mers pentru a recunoaşte intenţiile depinde de scopul pe care vrea să-l atingă agentul în procesul interactiv. În unele conversaţii - de exemplu când se negociază un plan de pace - este foarte important să se cunoască exact intenţiile partenerului atunci când vorbeşte. Alteori - cum ar fi când cineva te opreşte pe stradă şi te întreabă cât este ceasul - intenţiile interlocutorului nu sunt interesante (înafara celei evidente - vrea să ştie ora). Un sistem de operare computaţional trebuie să aibă specificat nivelul de detaliu necesar pentru aflarea intenţiilor; acesta va depinde de domeniul specific, de problema ridicată, de scopurile urmărite etc.

16.6 Recunoaşterea actelor ilocuţionare

După cum s-a subliniat anterior, comunicarea lingvistică nu apare decât dacă se recunoaşte intenţia de a face aceasta. Nu există însă o corespondenţă biunivocă între mulţimea frazelor şi cea a intenţiilor.

Astfel, am văzut că propoziţia

Ştii cât este ceasul ?

poate fi întrebare, o cerere sau o ofertă, în funcţie de ce a intenţionat vorbitorul. În acest caz, apare problema modului în care un agent recunoaşte intenţiile altui agent din spusele acestuia.

Cea mai uzuală soluţie la această problemă este bazată pe *ipoteza înţelesului literal*. Ea presupune că orice propoziţie are un înţeles bazat numai pe convenţiile limbajului. Astfel, exemplul de sus are următorul înţeles literal: propoziţia este o întrebare cu răspuns *Da/Nu* în care ascultătorul este chestionat dacă ştie cât este ceasul.

Acesta este numit *actul de vorbire de suprafaţă* şi este dat de modul sintactic al propoziţiei (aici, o propoziţie interogativă). Fiind dat înţelesul literal, *actul intenţional de vorbire* este derivat printr-un proces de inferenţă, cum ar fi cel de recunoaştere a planurilor.

Sunt două moduri de a privi actul de vorbire de suprafaţă. În prima variantă, acesta este un act ilocuţionar - care poate fi uneori greşit.

Astfel, pentru a pune în mod sincer o întrebare, vorbitorul trebuie să vrea să ştie răspunsul.

Exemplul 16.8 *Să considerăm o situaţie în care Elena ştie că Paul are probleme cu timpul, şi spune (ca o ofertă pentru ajutor):*

Ştii cât este ceasul ?

Actul de vorbire de suprafaţă este o întrebare cu răspuns dicotomic dacă Paul ştie cât este ora; el este ilocuţionar greşit deoarece Elena nu este interesată de răspuns.

Deducţia că actul de vorbire de suprafaţă este greşit, iniţiază căutarea pentru actul intenţional de vorbire. Din păcate nu este suficient ca actul de suprafaţă să fie greşit pentru a ca propoziţia să aibă un înţeles ascuns. Astfel, în exemplul de sus, *Elena* poate să nu ştie dacă Paul ştie cât este ceasul, şi întrebarea ei are două scopuri - acela de a primi un răspuns, dar şi cel referitor la faptul că ea se oferă să îl ajute pe Paul.

În cealaltă modalitate, actul de vorbire de suprafaţă nu este un act ilocuţionar ci un act la alt nivel de analiză, şi inferenţa trebuie să identifice actul intenţional de vorbire. Aici actul de vorbire de suprafaţă este doar o parte a analizei semantice a propoziţiei.

Revenind la contextul din exemplul anterior, întrebarea *Elenei* este o acţiune *Interog* cu conţinutul propoziţional

Şti(Paul1,timp).

Atunci, fiecare din actele ilocuţionare vor specifica ce forme de suprafaţă pot fi folosite pentru a-l realiza.

Figura 16.5 defineşte actul ilocuţional *CereRef* cu mai multe descompuneri posibile, fiecare corespunzând unei anumite forme de act de vorbire de suprafaţă. Mai este prezentat şi *MotivatInformRef*, care este o versiune specializată a acţiunii *MotivatDe Cerere.*

Cele trei descompuneri corespund formelor de suprafaţă folosite de obicei pentru transformarea unei acţiuni *CereRef.*

Chiar dacă se trece direct la citirea indirectă (omiţând citirea literală), forma propoziţiei poate afecta tipurile de răspuns. Aceasta sugerează că în analiza finală nu se poate face abstracţie de forma înţelesului literal.

Figura 16.5:

Clasa de acţiune:	**MotivatInformRef**
Roluri:	*Vorbitor, Auditor,* $Pred_x$
Restricţii:	*Agent(Vorbitor), Agent(Auditor), Pred_unar(*$Pred_x$*),*
	¬ŞtiRef(Vorbitor, $Pred_x$*), ŞtiRef(Auditor,* $Pred_x$*)*
Precondiţii:	*La(Vorbitor, Loc(Auditor))*
Efecte:	*Intenţie(Auditor, InformRef(Auditor, Vorbitor,* $Pred_x$*))*
Descompunere:	*CereRef(Vorbitor, Auditor,* $Prop_x$*)*
	Decide(Auditor, InformRef(Auditor, Vorbitor, $Pred_x$*))*
Act ilocuţionar:	**CereRef**
Descompunere_1:	*Ce(Vorbitor, Auditor,* $Pred_x$*)*
Descompunere_2:	*Interog(Vorbitor, Auditor, ŞtiRef(Auditor,* $Pred_x$*))*
Descompunere_3:	*Imperat(Vorbitor, Auditor,*
	InformRef(Auditor, Vorbitor, $Pred_x$*))*

Exemplul 16.9 *Să considerăm propoziţiile:*

Ştii cât este ceasul ?

Spune-mi ce oră este.

Câteva răspunsuri posibile la ele sunt:

a) Da, este ora trei.

b) Nu.

c) Ora trei.

d) Nu pot.

Răspunsurile (a, b, c) sunt normale pentru prima întrebare, pe când (d) este forţat. În schimb, la a doua propoziţie, răspunsurile (a, b) par forţate, iar (c, d) sunt normale.

16.7 Nivele de planificare ale discursului

Multe teorii ale discursului se bazează pe un set fixat de relaţii între propoziţii; aceste relaţii, numite *relaţii ale discursului, relaţii retorice, jocuri conversaţionale, mişcări ale discursului* (în funcţie de autori) definesc structura unui discurs corect formulat. În consecinţă, identificarea lor este esenţială pentru înţelegerea propoziţiilor din discurs. Afirmaţia este valabilă mai ales în partea de generare a limbajului, unde sistemul necesită prelucrarea unei mari cantităţi de informaţie, folosind multe propoziţii. În astfel de cazuri, partajarea informaţiei in unităţi de mărimea unei propoziţii este considerată similară unui proces de planificare bazat pe un set de relaţii de discurs.

Exemplul 16.10 *Să considerăm o aplicaţie în care sistemul are o interfaţă cu o bază de date enciclopedică. Una din problemele pe care trebuie să le realizeze sistemul poate fi definirea claselor de obiecte. În mod normal, o astfel de definiţie poate lua mai multe propoziţii. Definirea unui set de acţiuni de nivel al discursului, care*

folosesc diverse moduri de definire ale claselor, poate fi realizată ca o problemă de planificare. Astfel, două acţiuni de nivel al discursului, necesare la definirea claselor de obiecte, sunt definite în Figura 16.6. Fiecare pas intermediar din DefiniClasa este

Figura 16.6:

Act comunicativ:	**DefiniClasa(e)**
Roluri:	*Vorbitor, Audotir, Clasă*
Descompunere:	*IdentificaSuperClasa(Vorbitor, Auditor, Clasa)*
	IdentificaProprietăţi(Vorbitor, Auditor, Clasa)
	DaExemplu(Vorbitor, Auditor, Clasa)
-	
Act comunicativ:	**IdentificaSuperClasa(Vorbitor, Auditor, Clasa)**
Roluuri:	*Vorbitor, Auditor, Clasa*
Restricţii:	*Agent(Vorbitor), Agent(Auditor), SubTip(Clasa, SuperClasa)*
Descompunere:	*ConvingPrinInform(Vorbitor, Auditor, SubTip(Clasa, Superclasa))*

construit în funcţie de alte acţiuni ale actului de vorbire, cum este IdentificaSuper-Clasa. Sistemul va folosi aceste definiţii pentru a ajuta la planificarea unei secvenţe de propoziţii care definesc obiectul. Astfel, pentru o bază corespunzătoare de date, la întrebarea

Ce este un câine ?

sistemul poate planifica următoarele acţiuni de vorbire:
Câinii sunt animale.
Ei au de obicei dimensiuni mici.
De exemplu, Grivei este un câine.

Pentru a se obţine rezultate mai aproape de aşteptări, este necesar să se stabilească mai multe strategii diferite pentru definirea claselor şi a restricţiilor din fiecare clasă; pentru fiecare problemă particulară se va selecta strategia adecvată. De exemplu, unele clase de obiecte sunt mai bine definite prin comparare şi contrast cu alte clase care sunt deja cunoscute.

Alte aplicaţii folosesc nivelul discursului pentru determinarea structurii dialogului.

Exemplul 16.11 *Să considerăm un sistem automat de comandă, angajat într-un dialog cu un utilizator pentru cumpărarea unui produs pe baza unui catalog.*

Aici vor trebui definite mai multe etape ale dialogului, cum ar obţinerea adresei şi ordinului de plată, verificarea ordinului, emiterea de informaţii privind modalitatea de livrare a mărfii. Un astfel de sistem poate fi coordonat de un discurs (mult simplificat) reprezentat în Figura 16.7

Evident, fiecare acţiune poate fi descompusă în continuare. De exemplu, VerificaNumar este compusă din cel puţin două scrieri: Prima - când numărul ordinului de plată este corect - foloseşte simpla bază de cunoştinţe a sistemului. A doua - când numărul nu este corect - va continua dialogul pentru reconfirmarea lui şi posibil, pentru a obţine mai multă informaţie, eventual alt număr.

Figura 16.7:

Discurs scris:	**IaComanda**
Roluri:	*Sistem, Utilizator, Nume, Adresa, OrdinPlataInf, Produse*
Descompunere:	*BunaZiua(Sistem, Utilizator)*
	DaAdresa(Sistem, Utilizator, Adresa)
	DaOrdinPlataInf(Sistem, Utilizator, OrdinPlataInf)
	DaComanda(Sistem, Utilizator, Produse)
	VerificaLivrare(Sistem, Utilizator, Produse)
	Inchide(Sistem, Utilizator)
–	
Discurs Scris:	**DaOrdinPlataInf**
Roluri:	*Sistem, Utilizator, OrdinPlataInf*
Descompunere:	*MotivaInfRef(Sistem, Utilizator,* $\lambda.x$ *OrdinPlataInf=* x*)*
	ConvingePrinInfRef(Utilizator, Sistem, $\lambda.x$ *OrdinPlataInf=* x*)*
	VerificaNumar(Sistem, Utilizator, OrdinPlataInf)

În aplicaţii mai generale ale dialogului, planificarea nivelului discursului devine importantă din alte motive. Dacă utilizatorul este abilitat să controleze fluxul dialogului, atunci sistemul va trebui să fie poată recunoaşte ce face acest utilizator. De exemplu, el trebuie să recunoască momentul când utilizatorul schimbă subiectul conversaţiei şi să fie capabil să discute despre noul subiect. Similar, el trebuie să fie apt să recunoască când utilizatorul cere informaţii suplimentare despre anumite aspecte ale dialogului precedent.

În acest fel, complexitatea nivelelor de dialog poate fi tot mai ridicată.

Anexa 1

Calculul cu predicate de ordinul I

1.1 Definirea *FOPC*

Definiţiile din această anexă prezintă formal un model de logică, construit pe baza noţiunilor din [11] şi [12], unde sunt realizate aplicaţii ale calculului cu predicate în domeniul lingvisticii respectiv teoria mulţimilor. Pentru informaţii suplimentare se mai pot utiliza de exemplu [3], [10].

Orice teorie axiomatică se dezvoltă în cadrul unui sistem formal constituit din următoarele grupe de elemente: *simboluri primitive* (obiecte indivizibile indiferent de aspectul extern sau de modul de specificare), *formule* (construcţii sintactice pe baza simbolurilor primitive şi a unor reguli precise), *axiome* (formule alese după anumite criterii şi cu un anumit grad de specificare) şi *reguli de inferenţă* (care permit deducerea de noi formule pornind de la axiome).

Calculul cu predicate de ordinul I (*FOPC* - **F**irst **O**rder **P**redicate **C**alculus) este un caz particular al calculului propoziţional. El foloseşte drept categorii sintactice *Term* (Termeni), *Pred* (Predicate) şi *Form* (Formule).

Simbolurile primitive din *FOPC* sunt:

- *Simboluri logice*:
  - *Conectori*: $\neg$ (negaţie), $\vee$ (disjuncţie), & (conjuncţie), $\Rightarrow$ (implicaţie), $\Leftrightarrow$ (echivalenţă).
  - *Cuantificatori*: $\exists$ (cuantificatorul existenţial), $\forall$ (cuantificatorul universal).
- *Simboluri auxiliare*: '(',')'.
- *Constante*: valori primitive (eventual indexate) peste un alfabet fixat. De exemplu: $a, beta, c_2, \ldots$
- *Variabile*: elemente neprecizate, notate cu litere mici (eventual indexate) de la sfârşitul alfabetului ($x, y, z_5, \ldots$).

Vom nota cu C, V mulţimea constantelor respectiv a variabilelor. Acestea sunt mulţimi (teoretic) infinite şi disjuncte. Prin definiţie, $Term = V \cup C$.

Un aspect fundamental - din punct de vedere al aplicaţiilor - îl constituie stabilirea domeniului $\mathcal{L}$ al *obiectelor* de studiu, domeniu din care vor lua valori variabilele; acesta poartă de obicei numele de ***univers al discursului***. Evident, $C \subset \mathcal{L}$.
Tot aici vor fi definite şi mulţimile de valori ale predicatelor.

Predicatele sunt aplicaţii $\alpha : Term \rightarrow \mathcal{L}$. Vom nota cu $Pred_k$ mulţimea predicatelor α de aritate $k (k \geq 1)$. Astfel, $\in$ (apartenenţa), $=$ (egalitatea) sunt aplicaţii din $Pred_2$ în teoria mulţimilor; în lingvistică putem exemplifica cu
merge, mănâncă $\in Pred_1$, *sărută, iubeşte, arată* $\in Pred_2, \ldots$
Vom nota cu $Pred$ clasa tuturor predicatelor, indiferent de aritate.

Se defineşte inductiv clasa $Form$, astfel:

- $Term \subset Form$;
- Dacă $P \in Pred_n$, $T_1, \ldots, T_n \in Term$, atunci $P(T_1, \ldots, T_n) \in Form$;
- Dacă $P, Q \in Form$ atunci
  $$\neg P,\ (P \vee Q),\ (P \& Q), (P \Rightarrow Q),\ ((P \Leftrightarrow Q) \in Form;$$
- Dacă $P \in Form$, $x \in V$, atunci
  $$\exists x\ P \in Form, \quad \forall x\ P \in Form;$$
- Orice formulă se obţine prin aplicarea de un număr finit de ori a regulilor precedente.

Observaţii:

1. Uneori se pot introduce predicate noi pentru scrierea simplificată a formulelor. Astfel, $\neg(x = y)$ poate fi scrisă $x \neq y$ prin introducerea unui nou predicat binar $\neq$. Sau, în lingvistică, este cazul antonimelor.

2. Atunci când este posibil, parantezele sunt eliminate. Aceasta se poate realiza în funcţie de prioritatea conectorilor şi cuantificatorilor. Dacă $\pi(x)$ este prioritatea lui x, atunci
$$\pi(\neg) = \pi(\exists) = \pi(\forall) > \pi(\&) > \pi(\vee) > \pi(\Rightarrow) = \pi(\Leftrightarrow).$$

1.2 Axiomele *FOPC*

1. $A \Rightarrow (B \Rightarrow A)$
   $(A \Rightarrow B) \Rightarrow ((A \Rightarrow (B \Rightarrow C)) \Rightarrow (A \Rightarrow C))$

2. $A \Rightarrow (B \Rightarrow A \& B)$
   $A \& B \Rightarrow A$
   $A \& B \Rightarrow B$

3. $A \Rightarrow A \vee B$
   $B \Rightarrow A \vee B$
   $(A \Rightarrow C) \Rightarrow ((B \Rightarrow C) \Rightarrow (A \vee B \Rightarrow C))$

4. $(A \Rightarrow B) \Rightarrow ((A \Rightarrow \neg B) \Rightarrow \neg A)$
   $\neg\neg A \Rightarrow A$

5. $(A \Rightarrow B) \Rightarrow ((B \Rightarrow A) \Rightarrow (A \Leftrightarrow B))$
   $(A \Leftrightarrow B) \Rightarrow (A \Rightarrow B)$
   $(A \Leftrightarrow B) \Rightarrow (B \Rightarrow A)$

6. $\forall x(A(x)) \Rightarrow A(a)$

7. $A(a) \Rightarrow \exists x(A(x))$

Primele cinci axiome sunt valabile pentru orice calcul propoziţional, iar ultimele două sunt specifice *FOPC*.

Aceste axiome sunt de fapt *scheme de axiome*, în sensul că fiecare caz particular de alegere a formulelor conduce la o axiomă. De fapt, după cum se observă din modul de scriere, orice axiomă este o formulă (un element din *Form*).

1.3 Regulile de inferenţă ale *FOPC*

1. *Modus Ponens*:
$$\frac{A, A \Rightarrow B}{B}$$

2. *Universalizarea consequentului*:
$$\frac{C \Rightarrow A(x)}{C \Rightarrow \forall x(A(x))}$$
unde C nu conţine variabila x.

3. *Universalizarea antecedentului*:
$$\frac{A(x) \Rightarrow C}{\exists x(A(x)) \Rightarrow C}$$
unde C nu conţine variabila x.

Dacă regula *Modus Ponens* este adevărată în orice calcul propoziţional, ultimele două reguli de inferenţă sunt specifice *FOPC*.

După cum se vede, o regulă de inferenţă este un procedeu de deducţie a unei formule din altă formulă.

Teoremele (propoziţiile, lemele) sunt deduse din axiome folosind regulile de inferenţă. În acest context, se numeşte **demonstraţie** a unei formule $P \in Form$ orice secvenţă finită

$$\alpha_1, \alpha_2, \ldots, \alpha_n \qquad \alpha_i \in Form$$

definită astfel:

- $\alpha_n = P$
- α_i $(1 \leq i \leq n)$ este o axiomă sau se obţine din $\alpha_1, \ldots, \alpha_{i-1}$ prin folosirea unei reguli de inferenţă.

În aceste condiţii, formula P este *demonstrabilă*.

O axiomă este *independentă* de celelalte axiome dacă ea nu poate fi dedusă din ele.

Sistemul se numeşte *independent* dacă fiecare axiomă este independentă de celelalte.

Dacă pentru orice formulă P a sistemului, ori P ori $\neg P$ este demonstrabilă, atunci sistemul este numit *complet*; altfel, este *incomplet*.

Sistemul este *consistent* dacă nu există formule P astfel încât atât P cât şi $\neg P$ să fie demonstrabile.

Cuprins

Bibliografie

[1] Allen, J. - Natural Language Understanding, The Benjamin/Cummings Publ. Co. Inc, 1995

[2] Atanasiu, A - Bazele Informaticii; suport de curs pentru anul II seral; Tipografia Universitătii, 1987;

[3] Atanasiu, A. - Modele matematice în scrierea compilatoarelor; Ed. Olimp, 1996.

[4] Crăciun, D. - Analizor morfologic pentru limba română, Lucrare de licenţă, 1997

[5] Holan, T, Kubon, V, Platek, M - An implementation of Syntactic Analysis of Czech, Proceedings of Fourth International Workshop on Parsing Technologies, Prague and Karlovy Vary, sept. 20-24, 1995, pp. 126-135.

[6] Joshi, A.K., Levi, L.S., Takahashi, M. - Tree Adjunct Grammars, J. Comput. Syst. Sci. 10(1), 1975.

[7] Joshi, A.K. - How much context-senzitivity is necessary for Characterizing Structural description - Tree Adjoining Grammars; în "Natural Language Processing - Theoretical Computational and Psychological Perspectives", Cambridge University Press, 1985.

[8] Joshi, A.K. - The relevance of Tree Adjoining Grammar to generation; în "Natural Language Generation", Martinus Nijhoff Publishers, Dordrecht, 1987

[9] Schabes, Y. - Mathematical and Computational aspects of Lexicalized Grammars; Eight European Summer School in Logic, Language and Information, Praga, 12-23 August 1996

[10] Partee, B., Meulen, A., Wall, R. - Mathematical Methods in Linguistics, Kluwer Academic Publishers, 1990

[11] Partee, B., Borşev, V.B. - Fundamentele semanticii formale, Prelegeri seminarul Mathesus, Praga, martie 1998

[12] Ţiplea, F.L. - Introducere în teoria mulţimilor - Editura Universităţii "Al. I. Cuza", Iaşi, 1998

[13] Gramatica Limbii Române, Editura Academiei, 1966

Tiparul s-a executat sub cda 491/1998
la Tipografia Editurii Universității din București

ISBN 973 – 575 – 270 – 0

Lei 17100